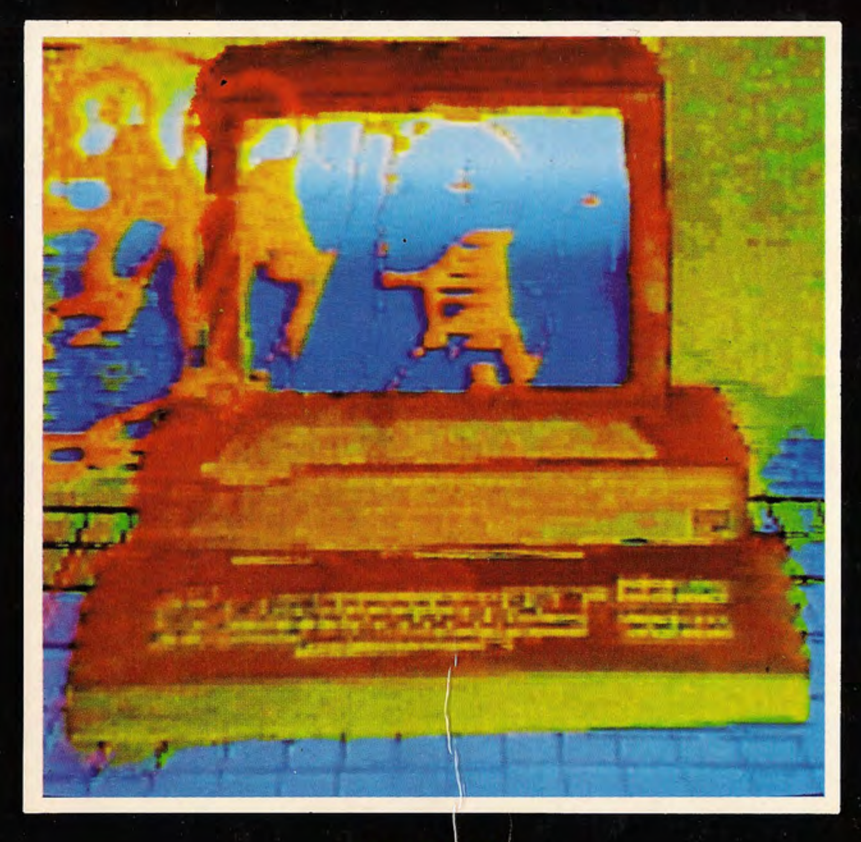
BIBLIOTECA BASICA INFORMATICA

DIBUJAR CON EL ORDENADOR

15



INGELEK

BIBLIOTECA BASICA

INFORMATICA

DIBUJAR CON EL ORDENADOR 15

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado.
Patricia Mordini.
Margarita Caffaratto.
Marina Caffaratto.
Francisco Ruiz.
Jorge Juan Monsalve.
Beatriz Tercero.
Fernando Ruiz.
Casimiro Zaragoza.

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-49-7
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-4868-1986
Precio en Canarias, Ceuta y Melilla: 380 pts.

INDICE

PROLOGO

En este libro nos ocuparemos de la creación de gráficos elementales con un ordenador personal. Al echar una rápida ojeada a todo aquello que normalmente se entiende por gráfico notaremos en seguida lo extenso que puede ser este tema: decir gráfico es decir dibujo, y dentro del dibujo está todo aquello comprendido entre los garabatos de un niño pequeño y las más famosas obras de arte; decir gráfico también es decir comunicación, y aquí el campo es enorme: fotografías, dibujo técnico, publicidad, murales... son ejemplos de cómo los gráficos son utilizados como vehículo de un mensaje. Y cuando, finalmente, pasamos de un dibujo estático al dinamismo de una película...

Desde el momento en que el ordenador adquiere la capacidad para realizar gráficos, hecho que se remonta a los años sesenta, la informática ha empezado a inmiscuirse en todo lo relacionado con ellos, sin dejar de lado ninguno de los temas mencionados anteriormente. Hasta los murales han evolucionado: potentes proyectores de luz láser, guiados como es lógico por un ordenador, son utilizados hoy en día para llevar a cabo inscripciones o dibujos (que pueden hasta estar en movimiento) sobre las más extrañas superficies: desde las paredes de una discoteca a las de una gran montaña e incluso en las nubes.

En el mundo del cine está aprovechándose cada vez más la posibilidad de sintetizar imágenes a placer, sin que sea necesario tener enfrente al sujeto al que se desea filmar; de esta manera ya no será necesario realizar costosas ambientaciones y, por otra parte, cualquier fantasía del director, por insólita que resulte, podrá ser llevada a la práctica. Baste recordar como ejemplo la película

"Tron", producida en los estudios Disney, en la que ha sido ampliamente utilizada la técnica de crear un gran número de imágenes por medio del ordenador para ser utilizadas luego en sucesión como si de una película corriente se tratara.

Los ejemplos mencionados hasta ahora poseen una cosa en común, y es que, en el fondo, toda imagen no es más que un símbolo trazado sobre una superficie cualquiera, si bien la naturaleza de estos dos elementos puede ser extremadamente variable, como es el caso del láser (en el que el signo no está constituido por materia). Al tratarse de gráficos en el ordenador, ambos elementos tendrán que ser compatibles con él: en lugar de tela y pincel nos encontraremos con que el espacio disponible será el de la pantalla, plotter o impresora gráfica, y el pincel se convertirá en el tubo catódico. Llegados a este punto es evidente que debemos hacer una selección, debido a la extensión del tema.

Dado que nuestro fin es esencialmente didáctico y de introducir en el tema, nos vamos a ocupar de la generación de imágenes por medio de un ordenador personal o doméstico, limitándonos a la pantalla de un monitor corriente. Esto supone que no hablaremos de la elaboración de imágenes en color sobre veloces y gigantescos ordenadores, ni de periféricos gráficos interesantísimos (como los plotters, por ejemplo), es decir, de nada que se halle fuera de las posibilidades de quien sólo posee un ordenador personal o doméstico. Nos limitaremos a las imágenes en blanco y negro y figuras esencialmente geométricas. Esto es, trataremos sobre las bases de gráficos computerizados desde el punto hasta el problema de las superficies escondidas. El resto, desde las obras de arte a las animaciones, vendrá en otra ocasión.

No vayan a creer que esto es poca cosa, pues, a pesar de que en este libro de la Biblioteca damos los elementos básicos de las Computer Graphics y de sus métodos (con numerosos y claros ejemplos, todos ellos en el tan popular lenguaje BASIC) nos hallamos aún lejos de haber tratado todo aquello que puede aparecer en la pantalla de un ordenador personal. Esto es sólo un comienzo: quienes se mantengan al tanto de las novedades editadas podrán hallar más "exquisiteces" sobre el tema de gráficos en ordenadores...

CAPITULO I

LOS GRÁFICOS VISTOS POR EL ORDENADOR

El monitor de vídeo: nuestro campo de batalla

El procedimiento según el cual se forma una imagen en la pantalla del tubo de rayos catódicos es siempre el mismo, ya sea tratándose de un televisor corriente o del monitor de un ordenador personal: un delgado haz de electrones, que proviene de un dispositivo llamado cañón electrónico, golpea la superficie interna de la pantalla desplazándose rápidamente de izquierda a derecha y de arriba abajo, recorriendo así una serie de líneas horizontales paralelas (Fig. 1).

La pantalla se halla revestida en su interior por los llamados fósforos, sustancias capaces de emitir luz cuando son "golpeadas" por los electrones del haz. Si el cañón dispara electrones durante el barrido constante, línea por línea, de toda la pantalla ésta aparecerá totalmente iluminada y, por tanto, blanca (o verde en el caso de un monitor de fósforo verde). Si, en cambio, el haz permanece quieto, la pantalla se presentará negra.

Ahora bien, ¿cómo pueden obtenerse imágenes algo más complejas e interesantes que una simple pantalla blanca o negra? Esto se logra al modular el haz, es decir, encendiéndolo o apagándolo continuamente de manera que vaya dibujando una imagen.

En el televisor la modulación del haz depende de la señal recibida por la antena, que transporta instante por instante la información correspondiente tomada por la cámara. En el caso del ordenador, la modulación del haz (y por tanto la de la imagen que resulta de ello) está bajo nuestro control y depende del contenido de cierta área de memoria. ¡No se le vaya a ocurrir desmontar

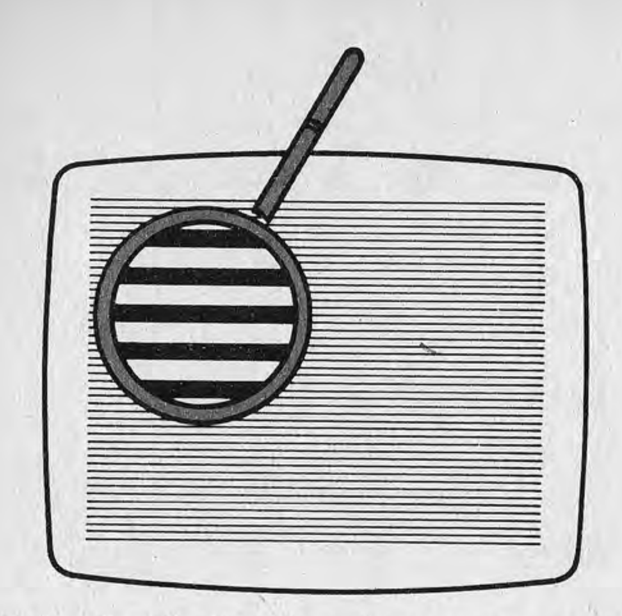


Figura 1.—El cañón electrónico se halla en constante movimiento y apunta hacia la pantalla recorriendo una serie de líneas horizontales. Cuando el cañón emite los electrones, las líneas se hacen luminosas y por tanto visibles. Cuando se efectúa el "retorno" (del final de una línea al principio de la siguiente) el cañón no emite y por ello estas líneas diagonales no se visualizan.

el monitor para observar el movimiento del cañón electrónico! En realidad, éste se produce al guiar el haz electrónico por medio de un campo magnético continuamente variable, sin que haya elementos en movimiento.

Parece ser, entonces, por lo visto hasta ahora, que la imagen se forma sobre una base constituida por una serie de líneas. En cambio, al observar atentamente la pantalla se notará que cualquier cosa representada está en realidad formada por una serie de puntos iluminados. En efecto, la imagen lograda depende precisamente del hecho de que estos puntos estén iluminados o apagados (Fig. 2). Hay ya en esto una diferencia en relación con lo que ocurre en un televisor corriente, en el que pueden aparecer imágenes en una posición cualquiera, sin que dependan de ninguna otra predeterminada por una red de puntos. Otra diferencia importante, con relación a la imagen de TV común, estriba en que un punto puede hallarse únicamente en dos estados posibles: iluminado o apagado, sin que exista posición intermedia. No se podrá, por tanto, visualizar ningún tipo de gris: todo tiene que ser

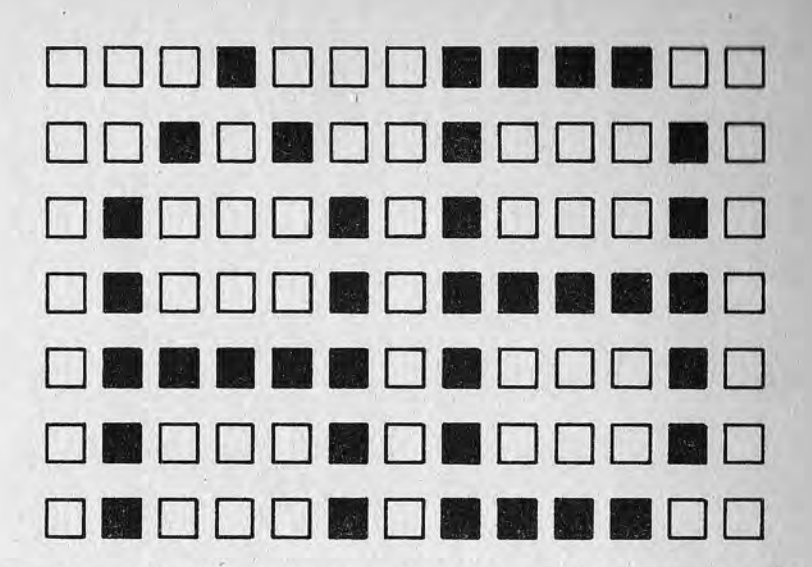


Figura 2.—La imagen generada por un ordenador está compuesta por puntos dispuestos en posiciones prefijadas, cada uno de los cuales puede estar iluminado o apagado. En la figura puede verse cómo aparecen las letras A y B observadas desde muy cerca; están señalados también los puntos que se hallan apagados.

blanco o negro, iluminado o apagado, 1 o 0. Que quede claro que estamos hablando de aparatos normales, pues en sistemas más sofisticados es posible visualizar diferentes tonos de gris y/o de colores; además, en el ordenador todo está cuantificado, grises y otros colores incluidos, de una forma reconducible a la binaria (sucesión de "unos" y "ceros").

Los modos gráficos

Si con el término gráfico entendemos todo aquello que puede representarse en la pantalla, el tema puede resultar demasiado extenso, por lo que será conveniente precisar un poco más.

Lo primero que vemos aparecer en la pantalla nada más encender el ordenador es, por regla general, una serie de caracteres alfanuméricos que normalmente indican el nombre del aparato y alguna información adicional. Lo que se está observando en ese momento es la llamada "Página de Texto": en ese momento en el monitor pueden aparecer cualesquiera caracteres elegidos entre el conjunto ("set") de los que hay disponibles.

De un ordenador a otro pueden variar tanto el número como el set de caracteres, según la configuración particular adoptada por el fabricante. Por ejemplo, el Apple visualiza normalmente 24 líneas con 40 caracteres cada una, pero añadiendo una tarjeta especial el formato puede resultar de 80 caracteres por línea, más útil para el caso en que se pretendan ejecutar trabajos de procesamiento de textos (Word Processing). También el Commodore 64 tiene una página de texto parecida: 25 líneas de 40 caracteres, pero el set comprende muchos caracteres que el Apple no posee (Fig. 3).

Juego 1	Juego 2	Decimal	Juego 1	Juego 2	Decimal	Juego 1	Juego 2	Decimal
@		0	V	v	22	,		44
A	a	1	W	w	23	–		45
B	q	2	X	x	24	.		46
C	c	3	Y	y	25	/		47
D	d	4	Z	z	26	0		48
E	e	5	[		27	1		49
F	f	6	£		28	2		50
G	g	7	]		29	3		51
H	h	8	↑		30	4		52
I	i	9	←		31	5		53
J	j	10			32	6		54
K	k	11	!		33	7		55
L	l	12	"		34	8		56
M	m	13	#		35	9		57
N	n	14	$		36	:		58
O	o	15	%		37	;		59
P	p	16	&		38	<		60
Q	q	17	'		39	=		61
R	r	18	(		40	>		62
S	s	19	)		41	?		63
T	t	20	*		42			64
U	u	21	+		43		A	65

Juego 1	Juego 2	Decimal	Juego 1	Juego 2	Decimal	Juego 1	Juego 2	Decimal
	B	66		W	87			108
	C	67		X	88			109
	D	68		Y	89			110
	E	69		Z	90			111
	F	70			91			112
	G	71			92			113
	H	72			93			114
	I	73			94			115
	J	74			95			116
	K	75			96			117
	L	76			97			118
	M	77			98			119
	N	78			99			120
	O	79			100			121
	P	80			101			122
	Q	81			102			123
	R	82			103			124
	S	83			104			125
	T	84			105			126
	U	85			106			127
	V	86			107			

Figura 3.—El set de caracteres es el conjunto de todos los caracteres que pueden aparecer en la pantalla o en la impresora. La forma de cada uno de ellos está establecida para cada ordenador y se encuentra almacenada generalmente de un modo permanente en ROM. El Commodore 64 posee también caracteres semigráficos.

En definitiva, independientemente de estas diferencias, todos los ordenadores personales tienen en común el hecho de poder funcionar de manera que presenten en la pantalla una página de texto. Esto último es cierto siempre que no se trate de aparatos específicos o donde el gráfico sea particularmente aparente, como es el caso del Macintosh.

Una característica específica de la página de texto es que los distintos puntos de la imagen no pueden ser iluminados o apagados de modo independiente, sino sólo en base a esquemas prefijados, es decir, según el set de caracteres y el "estilo" ("font") particular del ordenador. Esto significa que a pesar de que las letras y números son visualizados en forma de una matriz de puntos iluminados, con lo cual en teoría podrían representarse incluso, por ejemplo, todas las letras griegas, sólo se pueden obtener aquellos caracteres que ese ordenador en concreto contenga en su set (Fig. 3). Esto se debe a que normalmente el conjunto de los caracteres se halla en memoria ROM (sólo de lectura) de modo que no es posible alterarlo. Sin embargo, ciertos equipos permiten la obtención del set de caracteres a partir de una RAM (memoria de lectura y escritura); gracias a ello el usuario podrá representar cualquier carácter que desee, como ocurre, por ejemplo, en el Commodore 64.

Otra posibilidad de visualización que nos interesa de un modo particular es la "Página Gráfica". A diferencia con la página de texto, cuando el ordenador funciona de esta manera, cualquier punto de la pantalla puede ser iluminado o apagado individualmente. Es fácil imaginar las ventajas que esto conlleva: en la página gráfica es posible representar no sólo cualquier tipo de carácter, sino también cualquier dibujo que se desee. Para ello bastará con iluminar de forma correcta los puntos que lo componen, sin tener que depender para ello de un esquema fijo, como ocurre en la página de texto. Resulta evidente, por tanto, que para hacer gráficos computerizados es imprescindible que el ordenador pueda trabajar en el modo de página gráfica.

Existen además los llamados "sprites" (que significa literalmente "duende"), elementos gráficos definidos por el programador que pueden ser posicionados con facilidad en cualquier parte de la pantalla y moverse en ella. No todos los ordenadores tienen la capacidad de visualizar sprites, ni tampoco los manejan de la misma manera.

Aparte de detalles de menor importancia, como son el número y color, una diferencia importante que existe, por ejemplo, entre el Commodore 64 y los ordenadores de la serie MSX es que, aunque ambos poseen sprites, el primero obliga al programador a usar una serie de PEEK y POKE (o bien el lenguaje máquina) para utilizarlos, mientras los unidos al estándar MSX proporcionan una serie cómoda y racional de instrucciones BASIC especializadas en su gestión. Los sprites son utilizados sobre todo en programas de juegos, donde resulta esencial el movimiento rápido de los objetos que aparecen en pantalla.

Por último, vamos a citar los caracteres semigráficos que, en ciertos casos, pueden proporcionar prestaciones parecidas a las

que se obtienen en la página gráfica, si bien se utilizan sólo en la página de texto. En efecto, el set de caracteres de algunos ordenadores (Commodore, Spectrum y otros) no contiene sólo los habituales números, letras y signos de puntuación, sino también algunos símbolos formados por líneas, rectángulos y arcos de circunferencia; son los llamados caracteres semigráficos (o pseudográficos). Al componer adecuadamente tales símbolos es posible realizar algunos dibujos sencillos. Así, por ejemplo, utilizando los caracteres semigráficos del Commodore 64, que aparecen en la figura 3, ha sido hecho el dibujo de la figura 4.

Queda claro, en relación con lo dicho anteriormente, que en este libro no será posible profundizar en todos los modos gráficos indicados hasta ahora, ni en sus numerosas conexiones, pues nos extenderíamos demasiado. Nos ocuparemos, por tanto, de un modo particular de las bases geométricas de los gráficos computerizados, trabajando precisamente sobre la página gráfica.

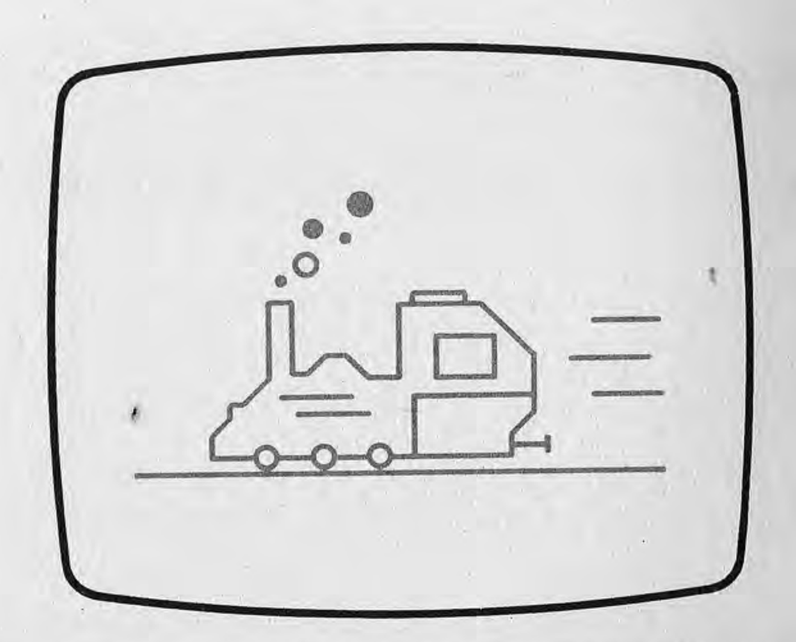


Figura 4.—Al componer de forma adecuada algunos caracteres semigráficos es posible obtener dibujos sencillos, como el que aparece en la figura.

Detalle y resolución

Todo lo que hemos comentado hasta ahora nos lleva a la formulación de una pregunta: la ventaja fundamental de la página gráfica hemos visto que es la de permitir el control de un único punto entre los existentes en la pantalla ahora bien, ¿cuál es el número de estos puntos? Como es fácil suponer, el tipo de dibujo que se puede hacer depende de que los puntos disponibles sean muchos o pocos. Además, según el trabajo que se quiera realizar, no será siempre necesario tener a nuestra disposición un número de puntos elevado.

Veamos una analogía que nos aclare el problema. Si, por ejemplo, estamos dibujando sobre papel, está claro que obtendremos resultados diferentes según utilicemos un lápiz de punta redonda u otro bien afilado. En el primer caso no será posible señalar detalles por debajo de cierto límite, mientras que en el segundo el grado de precisión, debido al tipo de punta, será mucho mayor. Este ejemplo es válido a la hora de comparar un ordenador en cuya pantalla aparezcan, digamos, 100×75 puntos en relación con otro capaz de visualizar 300×200 (cuando se dice 300×200 se entiende que sobre la pantalla aparecen 200 líneas, cada una de las cuales está formada por 300 puntos). Si las dimensiones de ambos monitores son las mismas está claro que en el primer caso el tamaño de los "puntos" será mayor que en el segundo.

En resumen, si se dispone de pocos puntos se obtendrán imágenes más bien bastas y tan sólo esbozadas; decimos en este caso que son "de baja resolución". Un mayor número de puntos permite, en cambio, una resolución más elevada, con detalles más precisos y concretos (Fig. 5).

La magnitud que define el grado de detalle alcanzable sobre una página gráfica es, precisamente, la resolución.

Una manera de medir la resolución puede ser, como hemos hecho anteriormente, refiriéndonos al número de puntos en que está dividido el espacio de la página gráfica tanto vertical como horizontalmente. Por ejemplo, el Apple posee un página gráfica cuya resolución es de 280×192, de modo que pueden encenderse y apagarse individualmente 53760 puntos (280×192). El Commodore 64 posee una resolución mayor: 320×200, es decir 64000 puntos.

Hay que tener en cuenta que al indicar la resolución el número más grande señala siempre el de los puntos de cada línea (horizontales, por tanto), mientras que el pequeño señala el número de líneas o, lo que es lo mismo, la cantidad de puntos verticales. Esto se debe a dos factores. Por un lado, la forma de la pantalla es la de un rectángulo cuyo lado mayor está dispuesto horizontalmente; al ser preferible tener unos puntos cuadrados o, al

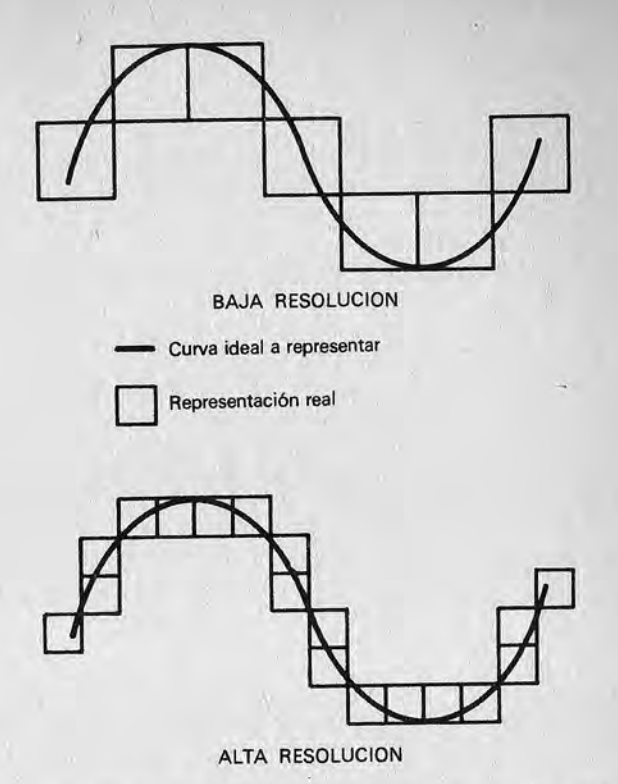


Figura 5.—El grado de resolución depende del número de puntos disponibles en la pantalla. Esto mismo determina las dimensiones de los puntos en sí.

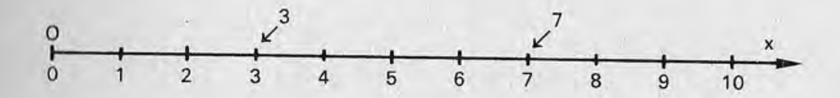
menos, lo más cuadrados posible, es evidente que debe haber un número mayor en el lado horizontal que en el vertical. Con más precisión, la relación entre los lados de la pantalla de un monitor (o de un televisor) es de 3/4. Hay además otro motivo que tiene que ver con los estándares de transmisión de las imágenes de TV, de tal manera que no es posible aumentar a placer el número de líneas (que corresponde con el número de puntos en vertical).

La base de todo aquello que puede ser representado en una página gráfica es el punto elemental que en el lenguaje de los gráficos por ordenador es llamado "pixel" (contracción de las palabras "picture element", es decir, elemento de la imagen). Del mismo modo que no es posible concebir una cantidad de información más pequeña que el bit, así en la pantalla de un ordenador gráfico no puede representarse nada cuyas dimensiones sean inferiores a las de un pixel.

El plano cartesiano

Una vez aclarado que nuestro campo de acción es una página gráfica con cierto número de pixels controlables individualmente, es necesario establecer un convenio que nos permita indicar sobre qué punto de la pantalla deseamos actuar. Esto resulta fácil debido a que los pixels se hallan en un plano, con lo que podremos utilizar las coordenadas cartesianas. Vamos a dar un repaso a este tema.

Haciendo referencia a la figura 6, para indicar la posición de un punto sobre una semirrecta "X" con origen en cero bastará con



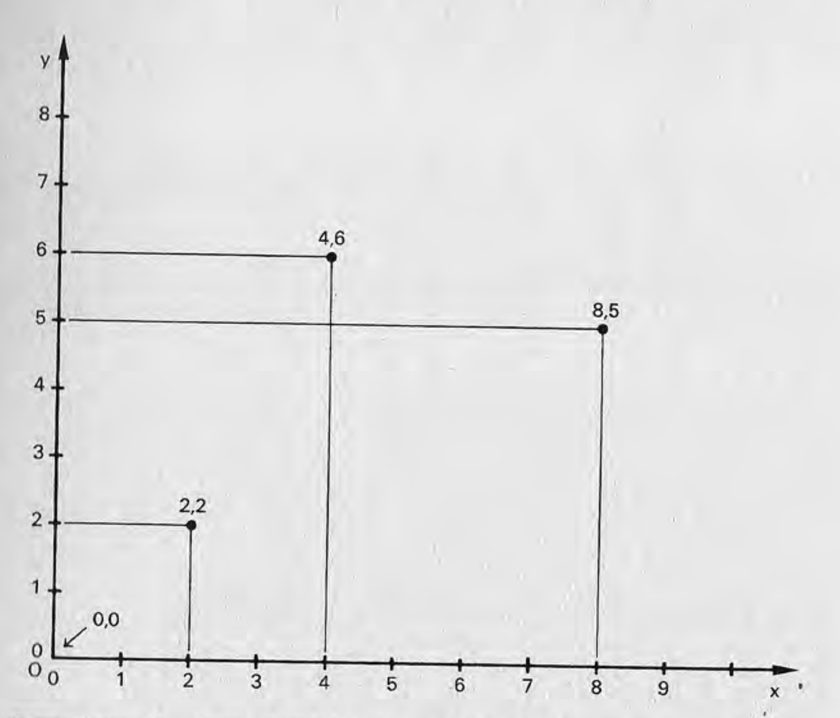


Figura 6.—Para identificar la posición de un punto sobre una semirrecta es suficiente un solo número. Si el punto se halla sobre un plano se requerirán dos, uno por cada una de las direcciones "X" e "Y". Cada pareja de números señala un único punto.

medir la distancia a ese punto desde el origen. Si, en cambio, el punto se halla en un plano delimitado por dos semirrectas, que llamaremos "x" e "y", serán necesarias las medidas de dos distancias para identificarlo: las que hay entre el punto y la semirrecta "x" y entre el punto y la semirrecta "y" (ambas medidas se toman perpendicularmente con relación a las dos semirrectas). Un plano en el cual los puntos se localizan por este convenio se llama plano cartesiano, y las semirrectas que lo definen, ejes.

Cada pixel de la página gráfica se identifica mediante una pareja de números llamada, precisamente, coordenadas cartesianas. Está establecido por convenio que el primero indique la distancia al eje "y", es decir, su posición horizontal (coordenada X) y el segundo al eje "x", que corresponde a la posición vertical (coordenada Y). En la figura 6 aparecen algunos ejemplos.

A pesar de que lo descrito anteriormente es el modo clásico de entender las coordenadas cartesianas, en la página gráfica de los ordenadores personales no siempre se sigue al pie de la letra tal convenio, aunque también es cierto que la diferencia es más que nada de carácter formal: el eje "y" está dirigido hacia abajo en vez de hacia arriba (en el caso del Apple, por ejemplo), mientras que el origen (con coordenadas 0,0) se halla en el ángulo superior izquierdo en vez de en el derecho. De ahí se deriva que las distancias positivas a lo largo del eje "y" se midan de arriba a abajo (Fig. 7).

¿Elegir un ordenador?

Este es, en principio, un libro esencialmente práctico que incluye incluso listados del software útiles para comprobar, paso a paso, las nociones aprendidas en cada momento. Pero se nos planteó un problema: ¿para qué ordenador iban a ser escritos estos programas? En efecto, para poder dar ejemplos concretos fue necesario hacer una elección dentro de la jungla de ordenadores que es el mercado.

Tras un complejo análisis hemos preferido en esta ocasión el conocidísimo Commodore 64. De todos modos, aquellas personas que posean otro ordenador, por ejemplo un Apple, un Spectrum o uno con sistema MSX, no tendrán dificultades a la hora de adaptar estos programas a su propio ordenador. Todo lo que se requiere es que éste posea las instrucciones necesarias para dibujar puntos y trazar líneas; una vez que haya intervenido en las zonas del programa relacionadas con estas operaciones (y en otras pocas que veremos más adelante) el resto del procedimiento será independiente del ordenador utilizado en casa caso. Para dar ánimos a los menos atrevidos diremos que todo el software que

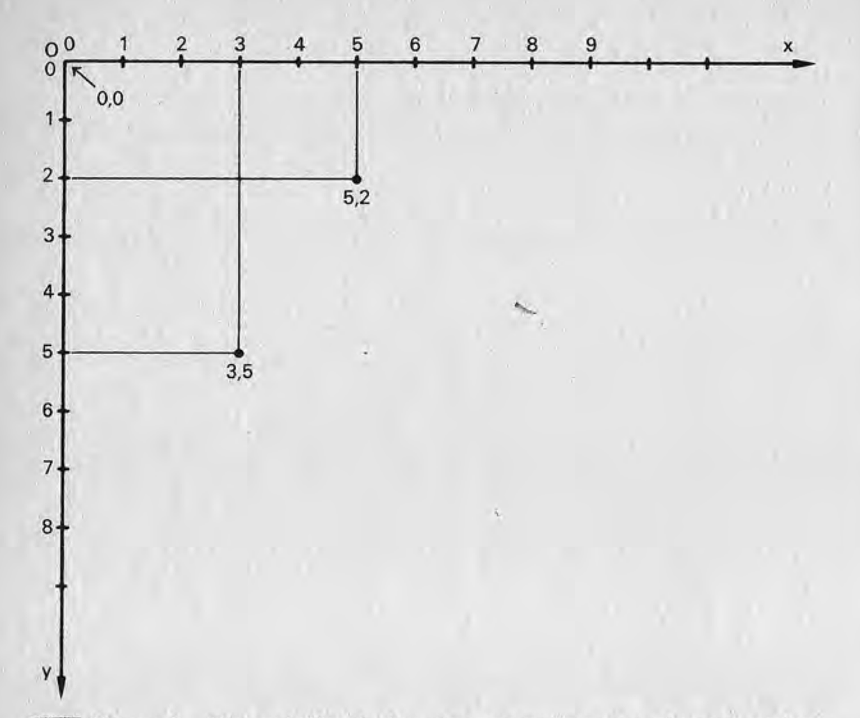


Figura 7.—Disposición de los ejes cartesianos en algunos ordenadores, con el origen situado arriba a la izquierda y el semieje "y" positivo hacia abajo. Este es el caso, por ejemplo, del Apple y del Commodore 64.

les vamos a presentar había sido inicialmente escrito para un Apple II; la readaptación ha sido, sin duda, pesada pero nada difícil. Si tienen cualquier tipo de problema en instrucciones que no sean de gráficos, no duden en repasar los volúmenes 5, 6 y 7 de la B.B.I. (especialmente el apéndice B de este último).

El primer obstáculo se ve enseguida: aquellas personas que tengan un Commodore 64 sabrán que éste no posee ninguna instrucción BASIC para gestionar la página gráfica. Para ello existe el BASIC Simon, una extensión del BASIC residente que añade algo así como 114 instrucciones nuevas, entre las que se hallan aquellas que nos interesan. Esta ampliación es bastante popular, de modo que quienes no la posean no tendrán dificultades para conseguir una copia (en disco o cinta). No está entre nuestros objetivos describir este BASIC, pues, por otra parte, existe un manual de la firma Commodore (por cierto, ¡es buena norma leer siempre por completo los manuales!); nos limitaremos sólo a considerar aquellas instrucciones útiles para nuestros fines...

Las instrucciones gráficas

En primer lugar, tras haber cargado el BASIC, habrá que dar al RUN con el fin de hacerlo operativo. La pantalla cambiará de color (si se trata de un monitor en color; en los demás cambiará sólo el tono de gris) y se pondrá blanca con el borde azul, mientras los caracteres aparecerán en negro. Dada la finalidad de este libro lo aconsejable es trabajar con el fondo negro y los bordes también en negro o, como mucho, de un color poco brillante. Para obtener el fondo y los bordes en negro habrá que teclear el comando:

COLOUR 0,0

en cambio, si se desea un contorno visible, por ejemplo de color azul celeste, habrá que escribir:

COLOUR 3,0

en ambos casos se deberá pulsar tras el comando la habitual tecla <RETURN>.

En esta situación los caracteres son visualizados en negro sobre fondo negro, con lo cual no son visibles, así que habrá que asignarles otro color diferente. Es aconsejable, para obtener la máxima legibilidad sobre fondo negro, trabajar con los caracteres en blanco: para conseguir esto bastará teclear un "control-white", que se obtiene manteniendo pulsada la tecla <CTRL> al tiempo que se aprieta la tecla <2>, que en la parte inferior lleva escrito precisamente "WHT". El color de los bordes no tiene nada que ver con las técnicas gráficas de que nos ocupamos, de modo que esta instrucción será utilizada sólo al comienzo del trabajo.

Por fin, tras haber limpiado el área de memoria con un NEW, estamos listos para comenzar. Tratemos ahora de las instrucciones gráficas en sí, de las que entre las varias que hay disponibles utilizaremos sólo cuatro:

— HIRES D,F
— PLOT X,Y,T
— LINE X1,Y1,X2,Y2,T
— NRM

Vamos a estudiarlas una por una.

HIRES D,F permite la visualización de la página gráfica y su borrado total, de manera que esté lista para el uso. D y F son dos números comprendidos entre 0 y 15, según una tabla (que se halla en el manual) que asigna uno de los 16 colores posibles a cada

número. El número D define el color que se desea utilizar para realizar el dibujo, mientras que F define el color de fondo. Para poner un ejemplo, la instrucción HIRES 1,0 (que veremos más a menudo) produce el efecto de borrar la página gráfica, poner un fondo negro y hacer que todo aquello que sea trazado en la pantalla sea de color blanco. Téngase en cuenta que el color de los bordes no es influido por esta instrucción.

PLOT X,Y,T es la instrucción que permite iluminar o apagar un solo punto. "X" e "Y" son las coordenadas características del pixel deseado. Como la resolución gráfica del Commodore 64 es de 320×200 puntos, "X" podrá variar entre 0 y 319, e "Y", entre 0 y 199 (los puntos están numerados a partir del cero). No hay que olvidar que el punto de coordenadas 0,0 (origen) se halla arriba a la izquierda, como ya vimos en la figura 7. El tercer parámetro de la instrucción PLOT, que hemos llamado T, sirve para establecer el tipo de operación, es decir, si se quiere iluminar o apagar el pixel indicado. El convenio establecido es el siguiente:

— 0 apaga el pixel de coordenadas X,Y
— 1 ilumina el pixel de coordenadas X,Y
— 2 invierte el estado del pixel de coordenadas X,Y; si antes de la instrucción estaba apagado se iluminará, si estaba iluminado se apagará.

LINE X1,Y1,X2,Y2,T permite trazar una línea (un segmento) sobre la página gráfica. Los parámetros X1,Y1 indican las coordenadas de un extremo del segmento; X2,Y2 las del otro. También en este caso habrá que dar otro parámetro T, con el mismo significado que posee en la instrucción PLOT, pero referido esta vez a toda una línea. Por ejemplo, con

LINE 0,0,100,50,1

se obtiene una línea diagonal que parte del origen y termina en el punto de coordenadas 100,50.

La última instrucción es NRM, que tiene como único efecto el volver a visualizar en la pantalla la página de texto en lugar de la gráfica. Es decir, con NRM se anula el efecto de HIRES.

Una observación importante: el BASIC Simon posee otras muchas instrucciones gráficas, algunas de las cuales resuelven por sí solas incluso algunos problemas que trataremos más adelante. El motivo por el cual no nos ocupamos de ellas es que esas instrucciones no son necesarias para comprender las bases de los gráficos computerizados, pues son a su vez "construidas" utilizando las dos instrucciones elementales de iluminar un punto y trazar un segmento. Precisamente estas dos son los verdaderos pilares de las técnicas gráficas, ya que al combinarlas es posible realizar cualquier operación que se desee. De esta manera, además, resultará mucho más fácil transportar nuestros programas a otros ordenadores, aunque tengan menos instrucciones potentes y particulares.

Transportabilidad de programas

El hecho de limitarnos a cuatro instrucciones gráficas elementales nos permite, como decíamos, poder "transportar" los programas de un ordenador a otro sin demasiados problemas. No es éste el lugar idóneo para examinar cada instrucción gráfica de los ordenadores más difundidos, pero daremos unas cuantas reglas generales:

- localizar cuáles son las instrucciones que poseen el mismo efecto que las que nos incumben. Si no las hubiera exactamente equivalentes será necesario estudiar qué diferencias existen, para poder más adelante modificar los programas de un modo adecuado. Si, por ejemplo, se tuviera que trabajar con un Apple, HIRES 1,0 se convierte en HGR2, NRM se traduce en TEXT mientras que PLOT y LINE se obtienen utilizando para ambas la instrucción HPLOT (para más detalles, consúltense los manuales). Compruebe además en qué ángulo se halla el origen de coordenadas;
- verifique si la resolución es la misma y, si no es así, modifique los programas de tal manera que se esté dentro de los límites (si se trata de un ordenador con menor número de puntos) o se pueda aprovechar al máximo el área disponible (en caso de que la resolución sea mayor);
- modifique los programas para adaptar la gestión de la página alfanumérica. No será necesario extenderse sobre este problema, que debería hallarse dentro de las posibilidades de cualquiera que conozca su ordenador;
- antes de modificar un programa para transportarlo a otro ordenador, estudie con atención la teoría que se halla en la base, de forma que se domine.

Otras indicaciones más específicas podrán hallarse en los capítulos en donde resulten útiles.

El software y sus "bugs"

Todos los programas que se hallan en el libro han sido probados anteriormente, pero todo programador sabe que los bugs

(fallos o errores, para que nos entendamos, pues su traducción literal sería "chinches") más insospechados se hallan siempre al acecho. Aunque cada programa resuelve un problema, esto no quiere decir que ésa sea la mejor manera de hacerlo. Este libro posee una función didáctica, por lo que alguna vez hemos preferido la claridad a una optimización extrema. Quien lo desee podrá tratar de perfeccionar las técnicas aquí utilizadas: buscar mejores algoritmos, utilizar trucos para aumentar la velocidad de ejecución, estructurar el programa de otra manera, etc. Esto, además, será un ejercicio estupendo para verificar el aprendizaje de las técnicas.

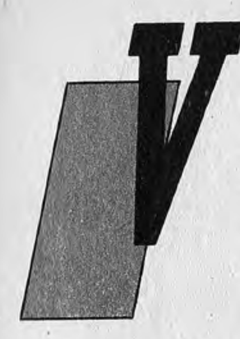
Según las características de esta Biblioteca, el presente libro ha sido proyectado para enseñar gradualmente algunos detalles más sofisticados y ciertas reglas de programación. Pretendemos ayudar a aquellos lectores que, todavía fresco su primer cursillo de BASIC, no están aún acostumbrados a escribir programas de una manera ordenada y comprensible.

Hablemos por último de los comentarios. Casi ningún programa los posee (con excepción de los más importantes) debido a que hemos preferido comentarlos abundantemente dentro del texto. De todas formas, siempre habrá un título con una numeración progresiva que indicará también el tipo de programa de que se trata.

CAPITULO II

GRÁFICOS UNIDIMENSIONALES

El reino de los puntos



Vamos a empezar con algunos programas que tratan de forma elemental los puntos y segmentos. El título de este capítulo es bien explícito: su contenido permanece en el ámbito de, como mucho, las líneas (aparte de alguna pequeña excepción) que, como es bien sabido, son lugares geométricos en una dimensión (de ahí que hayamos numerado en la forma "1.X" cada listado).

```
10 REM *** PROGRAMA 1.1 ***
20 REM
30 REM ILUMINA CUATRO PUNTOS
40 REM
50 REM
60 HIRES 1,0
70 REM
80 PLOT0,0,1
90 PLOT50,0,1
100 PLOT0,50,1
110 PLOT50,50,1
120 REM
130 GOTO 130
READY
```

El programa 1.1 tiene una función muy sencilla: la de encender cuatro puntos en la página gráfica, en correspondencia con los vértices de un cuadrado. A pesar de su simplicidad nos per-

mite ver en la práctica dos de la cuatro instrucciones gráficas citadas anteriormente, además de que podremos darnos cuenta de las dimensiones de los puntos.

Obsérvese también el orden en el que están dispuestos los distintos elementos del programa: en primer lugar hay un título con una breve descripción de las operaciones, después la instrucción que prepara el terreno visualizando la página gráfica y, por último, el cuerpo del programa. Estas tres secciones han sido idealmente separadas, de forma que resulte más fácil la lectura y la comprensión del listado; para este fin se ha utilizado una línea que contiene tan sólo un carácter (dos puntos), aunque puede haber otros tipos de separadores estéticamente válidos. Ciertamente estos detalles resultan muy útiles, sobre todo en programas de mayores dimensiones, pero siempre es mejor empezar a acostumbrarse a mantener cierto orden desde un principio.

```
10 REM *** PROGRAMA 1.2 ***
20 REM
30 REM ***  OSA MAYOR  ***
40 REM
50 REM
60 HIRES 1,0
70 REM
80 FOR I=1 TO 7
90 READX,Y
100 PLOTX,Y,1:PLOYX+1,Y,1
110 NEXT I
120 REM
130 DATA23,85,20,55,65,40,80,65
140 DATA115,60,145,63,180,50
150 REM
160 GOTO 160
READY
```

El programa 1.2 contiene una diferencia fundamental en relación con el anterior. El primero ha sido proyectado para crear una determinada imagen, que depende estrictamente del número de instrucciones PLOT (que son cuatro) y, sobre todo, del hecho de que las coordenadas de los puntos están expresadas en forma de constantes. El segundo, en cambio, visualiza una imagen que no depende tan específicamente del cuerpo del programa, sino más bien de lo que está contenido en una serie de DATA. El ciclo FOR...NEXT hace que la instrucción PLOT sea ejecutada siete veces, con coordenadas distintas cada vez.

La diferencia entre ambos programas puede parecer insignificante, pero no es así: con este programa podremos cambiar por

completo un dibujo sin tener que alterar sustancialmente el cuerpo del programa, sino sólo el límite del ciclo (el siete) y las líneas de DATA. En otras palabras, hemos construido un sencillo instrumento gráfico, que podremos utilizar para realizar funciones diferentes cada vez. De esta manera, la que ahora es la imagen de la Osa Mayor puede transformarse de una forma muy sencilla en cualquier otra figura con tal de que esté constituida por puntos: una constelación diferente, los vértices de un polígono, una imagen estilizada, etc.

Como en el ejemplo precedente, también en este programa hay una separación lógica entre las diferentes partes que lo constituyen: título, inicialización, cuerpo principal y datos para el funcionamiento.

Con el fin de ejercitarse en su comprensión y utilización puede ser interesante proyectar nuevas configuraciones de puntos para el programa 1.2. Nos podemos ayudar de una hoja de papel milimetrado sobre el que se hará el dibujo, después se determinarán las coordenadas de los puntos y se introducirán en las líneas de DATA. Cada pixel está representado por una pareja de coordenadas en el orden X,Y; el límite del ciclo se designará en función de su número. Dese cuenta que el resultado no depende de la secuencia con la que se han ido iluminando los distintos puntos.

Siguiendo aún dentro del tema astronómico, consideremos el próximo programa.

```
10 REM *** PROGRAMA 1.3 ***
20 REM
30 REM CIELO ESTRELLADO
40 REM
50 REM
60 HIRES 1,0
70 REM
80 FOR I=1 TO 100
90 X=RND(1)*320:Y=RND(1)*200
100 PLOTX,Y,1:NEXT I
110 REM
120 GOTO 120
READY
```

Se trata también de encender puntos, pero esta vez de forma aleatoria. Después de la inicialización, un ciclo FOR...NEXT se encarga de ejecutar cien veces las instrucciones que generan arbitrariamente una pareja de coordenadas, para iluminar luego el

pixel correspondiente. El resultado será, con un poco de suerte e imaginación, un cielo estrellado.

La generación de los valores de las coordenadas merece alguna atención, ya que hay una referencia a la resolución propia de la página gráfica del ordenador que estamos utilizando y éste será, por tanto, un punto sobre el que habrá que intervenir si se cambia de ordenador. Veamos, por ejemplo, la coordenada "X", que en el Commodore 64 debe estar comprendida entre 0 y 319. La expresión "RND(1)" genera un número aleatorio que se halla comprendido entre 0 y 1 (excluidos los extremos), de manera que al multiplicar esa cantidad por 320 (la resolución horizontal) se obtiene un número fraccionario que va desde 0.F hasta 319.F, indicando con F una parte fraccionaria cualquiera. Dado que la instrucción RND no devuelve nunca como valor un "1", sino siempre un número inferior a él, estamos seguros de que toda la expresión RND(1)*320 no valdrá nunca 320, logrando así el objetivo de mantenerse entre 0 y 319.

¿Y la parte fraccionaria? Pues bien, tanto el BASIC Simon como otros con capacidad gráfica, al hacer referencia a la página gráfica consideran solamente la parte entera de los números que son proporcionados como coordenadas. Por tanto, a pesar de ser conceptualmente correcto, resulta del todo inútil escribir

```
X=INT(RND(1)*320)
```

para obtener una coordenada "X" comprendida dentro del campo de variación aceptable (en este caso entre 0 y 319). Naturalmente, si se toma el límite 199 todo esto valdrá también en relación con la coordenada "Y".

Añadiendo refinamientos

Pero un verdadero cielo estrellado es muy diferente del que se obtiene con el método visto ahora. Puede mejorarse un poco el dibujo si las "estrellas" no están fijas en su brillo, sino un poco centelleantes, como ocurre realmente al mirar hacia el cielo.

Hacer parpadear un punto en la pantalla quiere decir encenderlo y apagarlo sin cambiar su posición, y esto implica que la posición sea conocida por el programa. Como el dibujo del cielo es casual, será necesario que las coordenadas de cada punto queden memorizadas por el programa en el momento en que es dibujada la imagen; posteriormente esas coordenadas pueden ser utilizadas para hacer centellear las estrellas, también de forma aleatoria.

Para este fin vamos a utilizar dos vectores, con un número suficiente de elementos como para contener las coordenadas de todas las estrellas. El núcleo del programa es casi idéntico al del precedente, con la única diferencia de que, además de dibujar las estrellas, es memorizada su posición en los dos vectores "EX" y "EY". Hacemos notar que, en caso de que se desee visualizar un cielo "más estrellado", además de aumentar el límite del ciclo se debería también actuar sobre el dimensionamiento de los vectores, aumentándolo.

```
10 REM *** PROGRAM 1.4 ***
20 REM
30 REM ESTRELLAS CENTELLEANTES
40 REM
50 REM
60 DIM EX(100),EY(100)
70 HIRES 1,0
80 REM
90 REM DIBUJO CIELO
100 REM
110 FOR I=1 TO 100
120 X=RND(1)*320:Y=RND(1)*200
130 EX(I)=X:EY(I)=Y
140 PLOTX,Y,1
150 NEXT I
160 REM
170 REM CENTELLEO ESTRELLA
180 REM
190 N=RND(1)*99+1
200 PLOTEX(N),EY(N),0:REM APAGA
210 FOR I=1 TO 20:NEXT I
220 PLOTEX(N),EY(N),1:REM ENCIENDE
230 GOTO 190
READY
```

Una vez dibujada la imagen, el programa procede a ejecutar el centelleo. En primer lugar se escoge una estrella, tomando un número aleatorio comprendido entre 1 y 100; a continuación esa estrella es apagada y en seguida encendida de nuevo. El proceso se repetirá así indefinidamente. Para obtener el número aleatorio, se saca en primer lugar uno entre 0 y 99 (con el método anterior) y luego se le añade uno.

El centelleo se produce en tres fases: el punto es apagado con

una instrucción PLOT que, en este caso, borra en vez de dibujar (ya que el último parámetro vale cero), a continuación es ejecutado un breve retraso por medio de un ciclo en vacío, y, por último, otro PLOT (con parámetro final uno) vuelve a encender la estrella.

A aquellas personas a las que les gusten los experimentos les aconsejamos que traten de modificar el proceso en una de las maneras siguientas:

— en lugar de encender y apagar la estrella desplácela ligeramente de su posición, de forma que se obtenga un efecto algo distinto, pero ¡cuidado con las que se hallan en los bordes!;
— aumente el realismo haciendo que centellee más de una estrella a la vez;
— ponga estrellas de diferente magnitud, como en un cielo de verdad.

Llevar a la práctica estos detalles requiere más práctica de la que ha sido necesaria hasta ahora; no resulta difícil llegar a hacerlo por uno mismo, pero no es motivo de preocupación el no lograrlo.

Encadenando puntos: los segmentos

Si en lugar de dibujar una serie de puntos casuales sobre la pantalla los situamos todos en fila, en una dirección determinada, obtendremos un segmento. En realidad, toda línea trazada sobre una pantalla gráfica está formada por puntos, de modo que las limitaciones debidas a la resolución de la pantalla también les afectan: en lugar de obtener una línea continua tendremos una "punteada". Cuanto mayor sea la resolución del ordenador sobre el que se trabaja más se parecerá el dibujo a un segmento (la Figura 1 ilustra claramente el concepto).

Ya que no es posible dibujar segmentos perfectos es necesario hacer otra observación que seguramente habrá usted pensado: el parecido con el segmento "original" dependerá también de su inclinación en relación con los ejes cartesianos. A igual resolución los mejores resultados se obtienen al trazar rectas verticales u horizontales, que no estarán nunca cortadas. Las líneas de 45 grados resultan también bastante aceptables debido a que, a pesar de estar muy cortadas, lo están de una manera uniforme. En cambio, los segmentos que poseen otras inclinaciones (en ambos sentidos, hacia el eje "X" o hacia el eje "Y") resultan más imperfectos, debido a las apreciables separaciones o saltos necesarios.

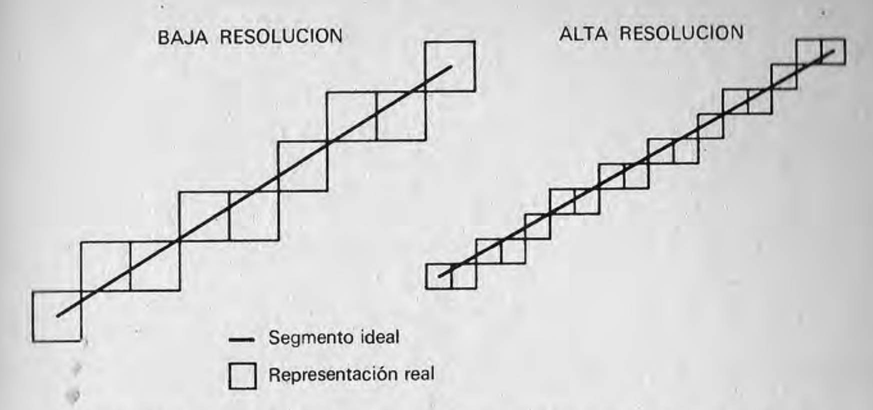


Figura 1.—La calidad de las líneas que pueden obtenerse sobre una página gráfica depende de su resolución, al estar formadas también por puntos. Una resolución más elevada (a la derecha en la figura) hace que el dibujo se parezca más al de una línea continua trazada con lápiz sobre un papel.

Veamos ahora cómo puede dibujarse una línea actuando sobre puntos individuales. Puede utilizarse para ello el programa 1.5, cuyo principio de funcionamiento está indicado en la figura 2.

```
10 REM *** PROGRAMA 1.5 ***
20 REM
30 REM LINEA FORMADA POR PUNTOS
40 REM
50 REM
60 REM PETICION VALORES
70 REM
80 INPUT "◘ COORDENADAS INICIALES.(X,Y)";XI,YI
90 INPUT "COORDENADAS FINALES.(X,Y)";XF,YF
100 INPUT "INCREMENTO";IN
110 REM
120 REM CALCULO
130 REM
140 LX=XF-XI:LY=YF-YI
150 L=SQR(LX**2+LY**2)
160 DX=LX/L:DY=LY/L
170 REM
180 REM TRAZADO LINEA
190 REM
```

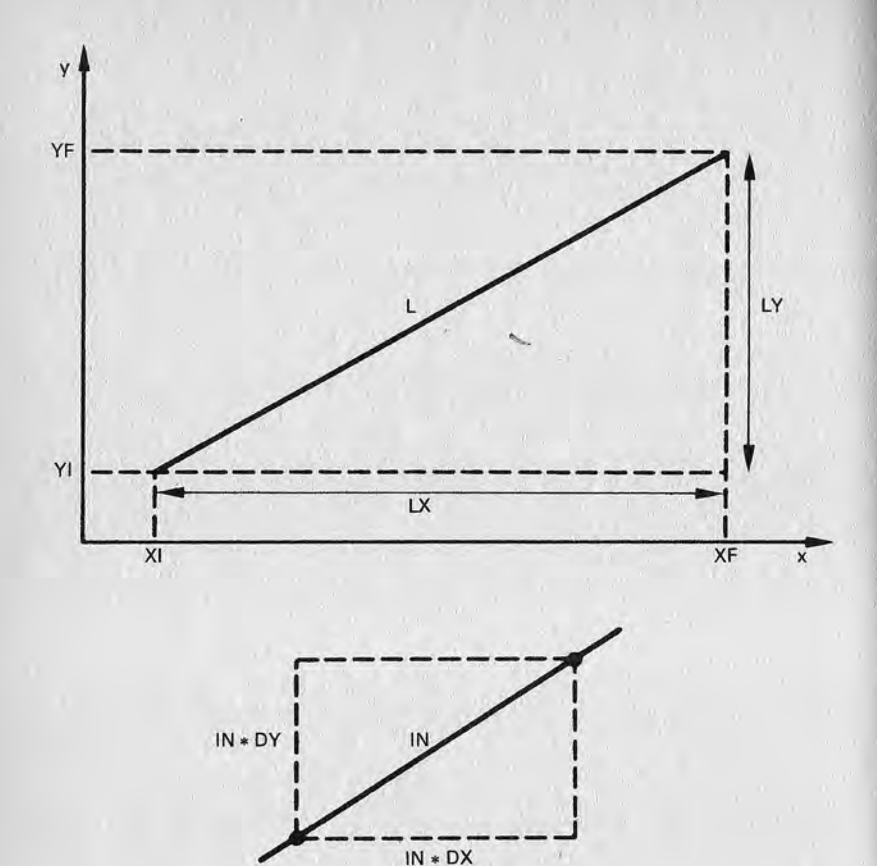


Figura 2.—En la figura se representan las relaciones geométricas entre las distintas variables del programa 1.5. En la parte inferior puede verse cómo a cada incremento "IN" en el segmento corresponde otro de las coordenadas "X" e "Y", que se obtiene para cada una al multiplicar IN por DX y por DY, respectivamente.

```
200 HIRES 1,0
210 FOR I=0 TO L STEP IN
220 PLOTXI+I*DX,YI+I*DY,1
230 NEXT I
240 REM
250 GET T$:IF T$="" THEN GOTO 250
260 NRM:GOTO 30
READY
```


El programa solicita las coordenadas iniciales y finales del segmento que hay que trazar y, como tercer dato, un valor del incremento. Haciendo referencia a la figura están justificadas las instrucciones de la sección cálculo, que no hacen otra cosa que obtener la longitud L del segmento aplicando el teorema de Pitágoras. Así, se calculan dos magnitudes (DX y DY), incrementos relativos de "X" e "Y" que expresan la relación entre la longitud del segmento y su proyección sobre los dos ejes cartesianos (LX y LY).

Todo esto será utilizado más tarde en la siguiente sección, que dibuja la línea. Para ello se recurre a un ciclo FOR...NEXT, que "mueve" el índice I a lo largo de todo el segmento que hay que trazar.

Supongamos que la variable IN vale uno; mientras el índice recorre paso a paso el segmento, como si se tratara de un lápiz sobre un papel, en cada ciclo del bucle se calcula la posición expresándola en las habituales coordenadas cartesianas referidas a los ejes. Esto se produce al sumar al origen del segmento, que es conocido (XI, YI), un desplazamiento obtenido utilizando cada vez los incrementos "DX" y "DY". La línea que se obtiene si se asigna el valor uno al incremento es un típico ejemplo de segmento trazado sobre una página gráfica. Puede hacerse una prueba dibujando otra línea con una inclinación diferente para comprobar los efectos, el mayor o menor parecido a un línea corriente, descritos anteriormente.

Tan sólo dos advertencias con relación al programa. A las preguntas acerca de las coordenadas debe darse una respuesta en la forma X,Y (por ejemplo 10,10 ó 137,42). La petición del incremento sirve para demostrar cómo este método puede ser utilizado para obtener líneas discontinuas: bastará responder con un número mayor que uno (con uno se obtendrá una línea "relativamente" continua). Otra observación importante es que no deben darse coordenadas correspondientes a un punto inicial o final que se hallen fuera de la pantalla, pues en ese caso el programa se parará indicando un error.

Para los no familiarizados con el Commodore 64 indicaremos que el carácter que aparece en la línea 80 tras las comillas equivale a la instrucción CLEAR (borrar pantalla). Para completar el significado de éste y otros símbolos que irán apareciendo en los listados vea el apéndice correspondiente.

Otro ejemplo de figura obtenida al combinar muchos puntos es el del rectángulo sombreado, dibujado por el programa 1.6. El programa traza una serie de segmentos horizontales, obtenidos siempre a base de puntos, hasta formar el rectángulo de las dimensiones deseadas, haciendo uso de dos ciclos FOR...NEXT anidados el uno en el otro.

```
10 REM *** PROGRAMA 1.6 ***
20 REM
30 REM AREA RECTANGULAR LLENA DE PUNTOS
40 REM
50 REM
60 REM PETICION VALORES
70 REM
80 PRINT " "
90 INPUT " ANGULO INFERIOR IZQUIERDO
   (X,Y)";X1,Y1
100 INPUT " ANGULO SUPERIOR DERECHO
   (X,Y)";X2,Y2
110 IF X2>=X1 AND Y1>=Y1 THEN 150
120 PRINT "   VALORES IMPOSIBLES":GOTO 90
130 REM
140 REM TRAZADO FIGURA
150 REM
160 HIRES 1,0
170 FOR Y=Y2 TO Y1
180 FOR X=X1 TO X2
190 PLOTX,Y,1:NEXT X,Y
200 REM
210 GET T$:IF T$="" THEN 210
220 REM
230 NRM:GOTO80
READY
```

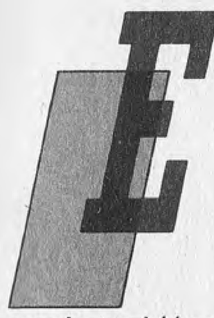
Este ejemplo muestra también de qué manera es posible ejecutar un control sobre los valores de las variables introducidas en el ordenador. Aquí el control no cubre todos los errores posibles de las entradas, pero demuestra igualmente su eficacia poniendo en evidencia (con un mensaje de error) aquellos valores que no pueden corresponder con los vértices solicitados.

Esta es una costumbre muy buena a la hora de programar, ya que elimina la posibilidad de que el trabajo en curso sea interrumpido por un error debido a quien se halla ante el teclado. Resulta muy útil sobre todo para programas complicados: un error causado al introducir valores fuera del límite previsto puede obligar a reanudar la ejecución, cosa que a veces puede llevar mucho tiempo.

CAPITULO III

GRÁFICOS BIDIMENSIONALES

Todo más cómodo con LINE



El dibujar una línea con el método de iluminar uno por uno todos los puntos que la componen no es nada corriente en gráficos computerizados. Como ya habíamos visto, entre las instrucciones fundamentales está la adecuada para trazar segmentos; hay que entender que los programas 1.5 y 1.6 son sólo ejemplos didácticos cuyo fin es el de mostrar el funcionamiento de un mecanismo por medio del cual el ordenador puede también visualizar segmentos en su página gráfica.

Una vez asimilado el principio, lo que se utilizará sin duda será la instrucción LINE específica para trazar líneas. En esta sección introduciremos precisamente una serie de programas que actúan sobre el plano para dibujar figuras compuestas por líneas, es decir, en dos dimensiones.

```
10 REM *** PROGRAMA 2.1 ***
20 REM
30 REM EJEMPLOS DE LINEAS
40 REM
50 REM
60 HIRES 1,0
70 REM
80 FOR Y=0 TO 180 STEP 20
90 LINE 0,0,300,Y,1:NEXT Y
100 REM
110 FOR X= 300 TO 0 STEP -20
```

```
120 LINE 0,0,X,180,1:NEXT X
130 REM
140 GET T$:IF T$=""THEN 140
READY
```

Estudiemos este programa. Muestra la capacidad del ordenador para trazar segmentos. En él aparecen dos ciclos FOR...NEXT: el primero une el vértice superior izquierdo de la pantalla (origen de coordenadas, ¡ojo!) con una serie de puntos del límite derecho, el segundo sigue la obra al unir el mismo vértice con otros puntos en el borde inferior. La figura que se obtiene sirve también para comprobar aquello que mencionamos anteriormente en relación con los dibujos de líneas inclinadas.

Con el Commodore 64 y el BASIC Simon puede observarse también otro detalle: las líneas verticales son menos gruesas que las horizontales. Este fenómeno, que resulta más evidente con un monitor, está causado principalmente por el limitado ancho de banda de la salida vídeo del Commodore, que no es capaz de guiar el cañón electrónico con una velocidad adecuada.

En efecto, ya dijimos que en una pantalla una recta vertical es dibujada durante el barrido de izquierda a derecha de cada línea como una serie de puntos individuales, iluminados uno tras otro en rápida sucesión. El hecho de que el ordenador y/o monitor no sean capaces de producir puntos perfectos influirá en toda la línea vertical. Pero esto mismo no es válido en relación con las líneas horizontales, que son trazadas por el cañón electrónico "de una sola vez". Es necesario mencionar el BASIC Simon porque otros BASIC pueden resolver el inconveniente simplemente utilizando como puntos para componer líneas verticales dos pixels contiguos (éste es el caso del Apple).

Un detalle curioso: si en lugar del monitor se utiliza un televisor doméstico, en general de inferior calidad, el inconveniente puede resultar menos evidente. Esto sucede precisamente porque el televisor tiene menor definición, así que no es posible distinguir los detalles más allá de un cierto límite, reduciendo de esta manera la diferencia visual entre una línea bien trazada y otra que no lo está.

```
10 REM *** PROGRAM 2.2 ***
20 REM
30 REM LINEAS ALEATORIAS
40 REM
50 REM
60 DEF FNX(A)=RND(1)*320
70 DEF FNY(A)=RND(1)*200
```



```
80 REM
90 INPUT "▣CUANTAS LINEAS";N
100 HIRES 1,0
110 REM
120 FOR I=1 TO N
130 X1=FNX(0):X2=FNX(0)
140 Y1=FNY(0):Y2=FNY(0)
150 LINE X1,Y1,X2,Y2,1
160 NEXT I
170 REM
180 GET T$:IF T$="" THEN 180
READY
```

El programa 2.2 dibuja en la pantalla algunos segmentos con orientación y longitud arbitrarios. El mecanismo es sencillísimo; la elección de los números aleatorios se produce con la función RND, de la que ya hablamos a propósito del programa 1.3. La novedad estriba, en cambio, en la manera de utilizarla: en lugar de ejecutarla en cada ciclo se ha preferido volver a llamarla por medio de las funciones definibles del BASIC, en este caso FNX(A) y FNY(A), respectivamente, para las coordenadas "X" e "Y". Esto tiene como único fin el de dar al programa una estructura más clara y hacer, por tanto, que sea más fácil de modificar. Observen que el parámetro con el que son llamadas las funciones FNX y FNY es cero, ya que en este caso no desarrollan ningún cometido (se trata de números aleatorios).

El siguiente programa (2.3) traza un cuadrado en la página gráfica.

```
10 REM *** PROGRAMA 2.3 ***
20 REM
30 REM CUADRADO
40 REM
50 REM
60 HIRES 1,0
70 NRM:INPUT "▣ VERTICE SUPERIOR IZQUIERDO
   (X,Y)";X,Y
80 INPUT "LADO=";L
90 GOSUB 210
100 REM
110 LINE X,Y,X+L,Y,1
120 LINE X+L,Y,X+L,Y+L,1
130 LINE X+L,Y+L,X,Y+L,1
```

```
140 LINE X,Y+L,X,Y,1
150 REM
160 GET T$:IF T$="" THEN 160
170 GOTO 180
180 REM
190 REM CONMUTA LA PANTALLA A GRAFICA
200 REM
210 VC=53248:REM VC=VIDEO CONTROLLER
220 CI=56576:REM CIA #2
230 POKEVC+17,PEEK(VC+17)AND239OR32
240 POKEVC+24,(PEEK(VC+25)AND15)OR8
250 POKECI+2,PEEK(CI+2)OR3
260 POKECI,PEEK(CI)AND252
270 POKEVC+17,PEEK(VC+17)OR16
280 RETURN
READY
```

No se trata ciertamente de una obra de arte tanto en gráfico como en programación, pero nos permitirá hacer algunas observaciones. La figura se define al especificar las coordenadas de un vértice (el que se halla arriba a la izquierda) y la longitud del lado (Fig. 1).

El hecho de determinar la posición de una figura a partir de la de un punto que le pertenece es una técnica que permite trazar con facilidad la misma figura en un lugar cualquiera de la página gráfica sólo con variar la posición de ese punto. De esta manera se acaba por considerarla un módulo gráfico, un patrón utilizable con otros módulos (iguales o diferentes) para crear dibujos más complicados. En este programa incluso es posible dibujar en la pantalla varios cuadrados sin tener que borrar los obtenidos anteriormente.

Trataremos con más detalle de la técnica de módulos gráficos en el programa 2.5.

Volviendo al programa 2.3, en él se hace uso de la instrucción NRM, que, como se recordará, permite volver a visualizar la página de texto. Esto es necesario porque el programa está proyectado para dibujar varios cuadrados y para cada uno de ellos es necesario pedir al operador su posición y dimensiones.

No resulta nada fácil descubrir por uno mismo el método con el que se vuelve a visualizar la página gráfica sin borrar su contenido anterior. El procedimiento lo hemos desarrollado en forma de subrutina por facilitar el seguimiento del programa. Se trata, básicamente, de manejar los registros del integrado de control de vídeo (VIC 6567) con el fin de provocar la visualización del área

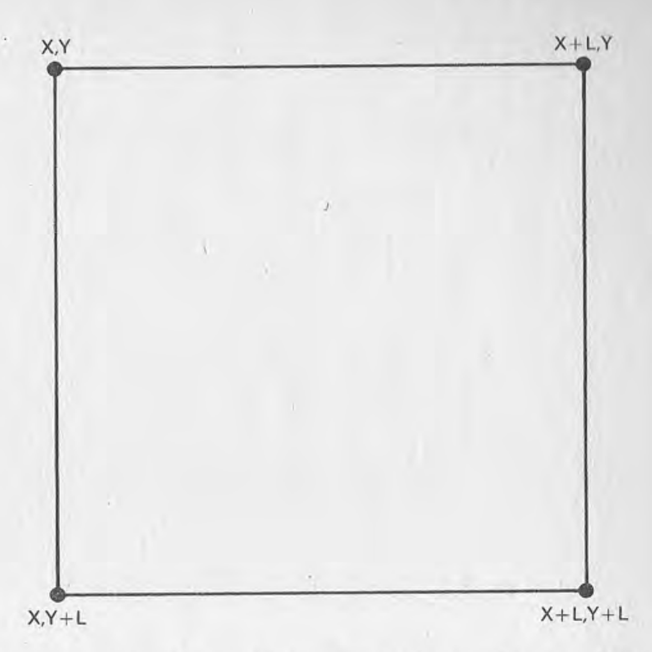


Figura 1.—Los vértices de un cuadrado pueden ser definidos por medio de las coordenadas de uno de ellos y la longitud del lado.

de memoria que constituye la página gráfica. Se entiende, por tanto, que la operación (regresar a la página gráfica sin borrarla) depende estrictamente del ordenador que estamos utilizando, así que será necesario revisarla en caso de que se quiera escribir un programa parecido para otro ordenador.

La última observación se refiere a que, con toda probabilidad, la figura no se parecerá mucho a un cuadrado. Aquí entran en juego las proporciones de la pantalla y las de la página gráfica, temas de los que hablaremos más adelante.

```
10 REM *** PROGRAMA 2.4 ***
20 REM
30 REM CUADRICULA
40 REM
50 REM
60 HIRES 1,0
70 REM
80 FOR Y=30 TO 100 STEP 10:LINE 30,Y,140,Y,1
   :NEXT Y
```

```
90 FOR X=30 TO 140 STEP 10:LINE X,30,X,100,1
   :NEXT X
100 REM
110 GET T$:IF T$="" THEN 110
READY
```

Antes de pasar a algo más complejo, veamos otro ejemplo de utilización de los segmentos, como el mostrado por el programa 2.4, que dibuja una serie de líneas en forma de cuadrícula. Esto es llevado a cabo por medio de dos ciclos FOR...NEXT que trazan, el primero las líneas horizontales y el segundo las verticales. En este caso resulta particularmente evidente el defecto que ya mencionamos con relación al grosor de los segmentos verticales, sobre todo si se usa un monitor. Se puede intentar mejorar la situación haciendo que las líneas verticales tengan doble anchura. Bastará para ello que en cada ciclo sean trazadas dos líneas contiguas en vez de una sola, añadiendo antes del "NEXT X" la instrucción

LINE X+1,30, X+1,100,1

Merece la pena realizar la prueba para darse cuenta de cómo es posible mejorar la calidad de la imagen tan sólo con hacer más visibles las líneas verticales.

Un pequeño proyecto gráfico: Manhattan

Nos vamos a sumergir ahora en un proyecto gráfico relativamente complejo en el que aparecerán muchos de los argumentos y técnicas tratados hasta ahora. Lo que queremos obtener es un dibujo estilizado que represente una serie de rascacielos (de ahí el nombre Manhattan) según un paisaje definido por el programa pero fácilmente modificable. Existe además la posibilidad de dibujar paisajes casuales, es decir, diferentes en cada momento.

Un ejemplo de lo que podrá verse en la pantalla se muestra en la figura 2. El programa 2.5 ha sido escrito teniendo en cuenta la exigencia de obtener un programa comprensible y modificable sin demasiadas dificultades. Vamos a resaltar algunos detalles para aquellas personas que estén dando sus primeros programas de gráficos computerizados:

- el listado está dividido en bloques funcionales (o secciones) que se reconocen con facilidad; cada uno de ellos está precedido por un breve título que especifica su función;

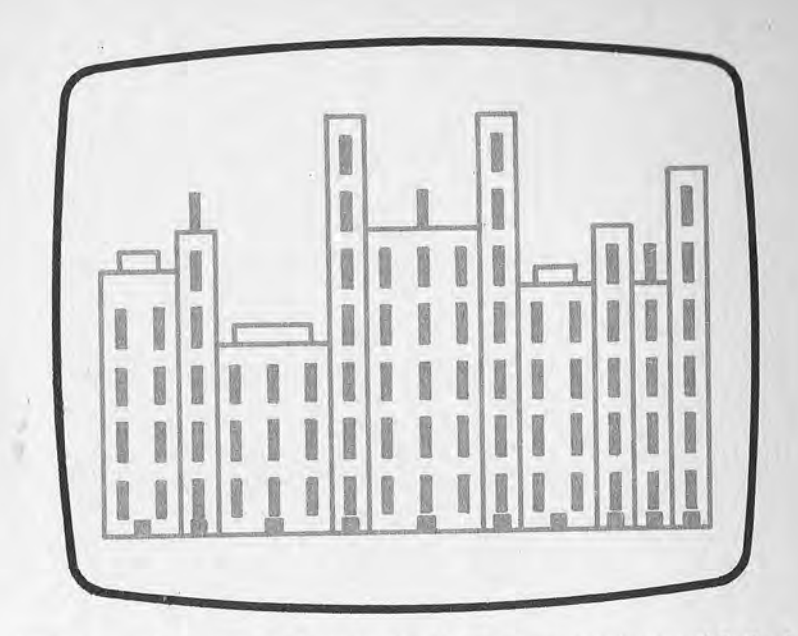


Figura 2.—Paisaje dibujado por el programa Manhattan (2.5). La forma de cada rascacielos se define fácilmente por medio de unos pocos parámetros contenidos en líneas de DATA.

- el dibujo de cada rascacielos es ejecutado siempre por la misma subrutina, llamada cada vez con parámetros diferentes. El significado de cada uno de los cuatro parámetros necesarios está especificado en los comentarios que se hallan al comienzo de la subrutina;
- gracias a este planteamiento el paisaje queda completamente definido con pocas constantes, insertadas en un grupo de líneas DATA, de tal modo que es muy fácil modificar el resultado final sin tener que tocar para nada el cuerpo del programa (ésta es una característica muy importante);
- cada rascacielos está constituido por la unión de diferentes elementos gráficos (muros, puerta, ventanas, terminación, ventanas de la terminación y punta) que no cambian su forma entre un rascacielos y otro (excepto los muros), sino sólo su posición y el que estén presentes o no;
- estos elementos han sido compuestos de forma que dependen tan solo de los cuatro parámetros antes mencionados, es decir: B (medida de la base), H (medida de la altura), T (indica la presencia de la terminación) y P (indica la presencia de la punta);

- todos los elementos están situados según una unidad de medida, que es la misma en todo el programa;
- hay ciertos controles que impiden que datos equivocados puedan provocar errores;
- una vez hecha la elección entre un paisaje definido u otro arbitrario, las secciones del programa que ejecutan las dos tareas están claramente diferenciadas entre sí.

Después de estos detalles no debería resultar difícil entender en su totalidad el funcionamiento del programa; aun así, examinaremos en detalle tanto el procedimiento utilizado como las relaciones geométricas entre los distintos elementos que constituyen cada rascacielos.

```
10 REM *** PROGRAM 2.5 ***
20 REM
30 REM MANHATTAN
40 REM
50 REM
60 RC=16:REM UNIDAD DE MEDIDA <RASC>
70 REM
80 OX=0:OY=199
90 PRINT "▣ PAISAJE PREDEFINIDO O CASUAL (D/C)";
100 INPUT R$:IF R$<>"D" AND R$<>"C) THEN 90
110 HIRES 1,0
120 IF R$="C" THEN 240
130 REM
140 REM PAISAJE PREDEFINIDO
150 REM
160 RESTORE:READ NR
170 FOR I=1 TO NR
180 READ B,H,T,P:GOSUB 540
190 OX=OX+B*NR
200 NEXT I:GOTO 340
210 REM
220 REM PAISAJE CASUAL
230 REM
240 B=INT(RND(1)*3)+1
250 IF OX+B*RC>320 THEN 340
260 H=INT(RND(1)*8)+3
270 T=RND(1)>.5
280 P=RND(1)>.5
290 GOSUB 540:OX=OX+B*RC
300 GOTO 240
310 REM
320 REM ESPERA UNA TECLA
330 REM
340 GET T$:IF T$="" THEN 340
350 NRM:PRINT "▣ OTRA VEZ (S/N)";:INPUT R$
360 IF R$<>"N" AND R$<>"S" THEN 350
370 IF R$="S" THEN 80
380 PRINT "▣▣▣ OK, ADIOS...":END
390 REM
400 REM ********************************
410 REM *                              *
420 REM *          SUBRUTINA           *
430 REM *     DIBUJO RASCACIELOS       *
440 REM *                              *
450 REM ********************************
460 REM
470 REM B=BASE
480 REM H=ALTURA
490 REM T<>0 =TERMINACION
500 REM P<>0 =PUNTA
510 REM
520 LINE OX,OY-H*RC,OX,OY,1
550 LINE OX,OY,OX+B*RC-1,OY,1
560 LINE OX+B*RC-1,OY,OX+B*RC-1,OY-H*RC,1
570 LINE OX+B*RC-1,OY-H*RC,OX,OY-H*RC,1
580 REM
590 REM TRAZADO PUERTA
600 REM
610 X1=OX+B*RC/2-5:X2=OX+B*RC/2+4
620 LINE X1,OY,X1,OY-9,1
630 LINE X1,OY-9,X2,OY-9,1
640 LINE X2,OY-9,X2,OY,1
650 REM
660 REM TRAZADO DE VENTANAS
670 REM
680 FOR I9=1 TO H-1:FOR I8=0 TO B-1
690 X3=OX+I8*RC+6:Y3=OY-I9*RC
700 FOR I7=0 TO 3
710 LINE X3+I7,Y3,X3+I7,Y3-8,1
720 NEXT I7
```

```
730 NEXT I8,I9
740 REM
750 REM DIBUJO TERMINACION
760 REM
770 IF T+0 OR B=1 THEN 930
780 X4=OX+7:X5=OX+B*RC-8:Y4=OY-H*RC:
    Y5=OY-(H+1)*RC
790 LINE X4,Y4,X4,Y5,1
800 LINE X4,Y5,X5,Y5,1
810 LINE X5,Y5,X5,Y4,1
820 REM
830 REM TRAZADOS VENTANAS TERMINACION
840 REM
850 FOR I9=0 TO B-2
860 X6=X4+I9*RC+5:X7=X4+I9*RC+12:Y6=Y4-8
870 FOR I8=0 TO 3
880 LINE X6,Y6-I8,X6+2,Y6-I8,1:LINE X7-2,
    Y6-I8,X7,Y6-I8,1
890 NEXT I8,I9
900 REM
910 REM TRAZADO PUNTA
920 REM
930 IF P=0 THEN 990
940 X8=OX+B*RC/2-2
950 Y8=OY-H*RC:IF T<>0 AND B>1 THEN Y8=Y8-RC
960 LINE X8,Y8,X8,Y8-3,1:LINE X8+1,Y8,X8+1,
    Y8-RC,1
970 LINE X8+2,Y8,X8+2,Y8-RC,1:LINE X8+3,Y8,
    X8+3,Y8-3,1
980 REM
990 RETURN REM * FIN SUBRUTINA *
1000 REM
1010 REM
1020 REM DESCRIPCION RASCACIELOS
1030 REM
1040 DATA 12:REM No.RASCACIELOS
1050 DATA 2,6,1,0,1,8,0,1,3,4,1,0,1,10,0,1
1060 DATA 3,6,0,1,1,10,0,1,2,5,1,0,1,8,0,0
1070 DATA 1,7,0,0,2,9,1,1,2,7,1,0,1,5,0,0
READY
```


Lo primero es definir una unidad de medida que podamos utilizar como estándar, con un valor cualquiera pero constante durante todo el programa. Al tratarse de rascacielos podemos llamarla "RASC" y su valor, fijado igual a 16 pixels, es asignado a la variable RC (línea 60). Las dimensiones del rascacielos quedan definidas por medio de la anchura de su base, indicada por la variable B, y de su altura H.

Veamos ahora de qué manera la subrutina dibuja un único rascacielos (desde la línea 400 a la 990); este proceso, repetido varias veces, creará luego el paisaje completo. Es importante hacer notar que el dibujo del edificio se lleva a cabo con referencia a un solo punto del mismo (criterio ya utilizado en el programa 2.3), que es en este caso el ángulo inferior izquierdo, es decir, el extremo izquierdo de la base. Este punto, definido por medio de sus coordenadas "OX" y "OY" (origen X y origen Y), se halla inicialmente a la izquierda en el fondo de la pantalla y se va desplazando hacia la derecha a medida que van dibujándose los diferentes rascacielos. La primera asignación de la posición del origen se produce en la línea 80.

El dibujo de cada rascacielos se efectúa empezando por su contorno exterior, ejecutado por la sección del programa llamado "TRAZADO MUROS" (líneas 520-570). Supongamos que queremos dibujar un rascacielos con base 2 RASC y altura 5; la relación entre sus vértices está indicada en la figura 3. En realidad, estas relaciones son bastante sencillas: se trata tan sólo de hallar la posición de tres vértices de un rectángulo dada la posición de uno de ellos abajo (a la izquierda, que es el origen) en base a la unidad de medida RC y a los valores de B y de H (base y altura).

Nótese que el extremo derecho de la base posee una coordenada igual a OX+(B*RC)−1. Ese "menos uno" sirve para que la longitud de la base sea precisamente igual al número especificado de RASC. En nuestro ejemplo la base mide 2 RASC, es decir, 32 pixels (1 RASC = 16 pixels). La coordenada "X" del extremo derecho de la base, que está dada por OX+(2*16)−1, vale OX+31; la del extremo izquierdo vale, por definición, OX; en total son, efectivamente, 32 puntos. El hecho de corregir esta unidad es necesario para dibujar la base, pues de otro modo los rascacielos ocuparían más espacio en horizontal de lo previsto.

En el caso de la altura tal corrección no ha sido aplicada para hacerlo más sencillo; lo único que ocurre es que la altura de los edificios aumentará en un pixel. Esto equivale a añadir a cada rascacielos un techo de un espesor de un punto, cosa que no entorpece para nada el dibujo.

El siguiente paso es el dibujo de la puerta. Este siempre se halla a nivel del suelo. Su posición se calcula fácilmente con sólo referirnos al origen, según los criterios que aparecen en la figu-

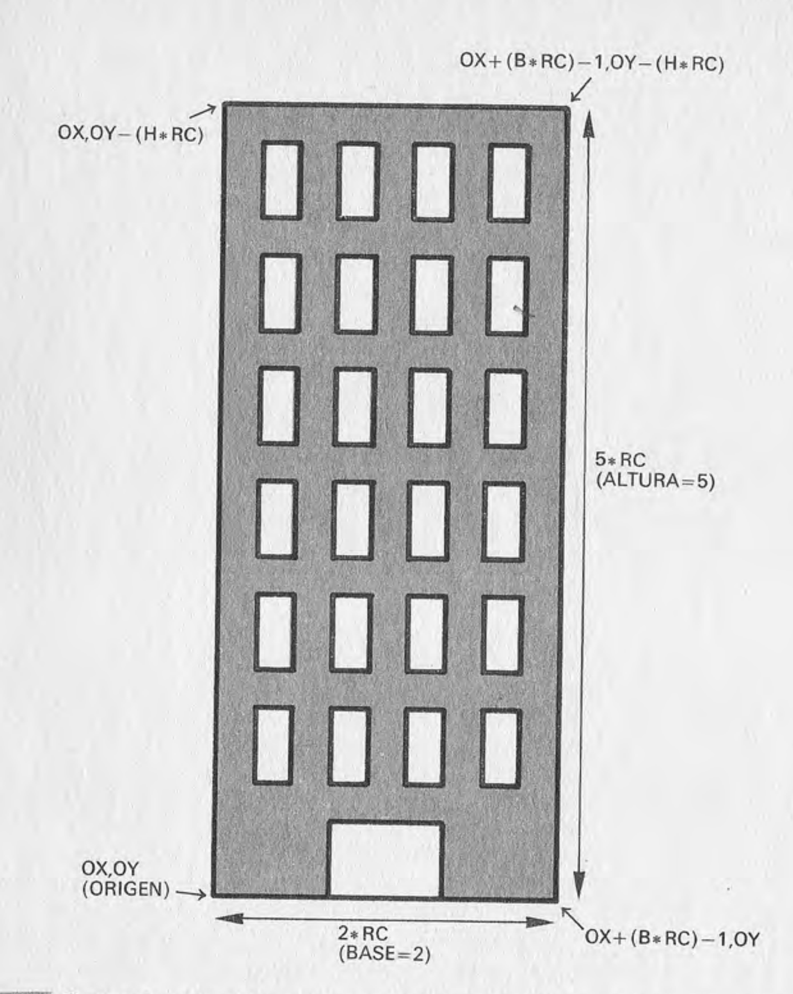


Figura 3.—Las relaciones geométricas entre los vértices de los muros del rascacielos están deducidas a partir de su origen, definido arbitrariamente como el extremo izquierdo de la base, de la longitud de ésta y de la altura.

ra 4, en donde en este caso se supone que la base del rascacielos posee una anchura de un RASC. No resulta difícil comprobar qué la puerta ha sido definida de tal manera que se halle siempre en posición centrada en relación a la base, independientemente de que ésta posea un número par o impar de RASCs. El segmento de programa llamado "TRAZADO PUERTA" (líneas 590-640) se encarga en primer lugar de calcular la posición horizontal de los pun-

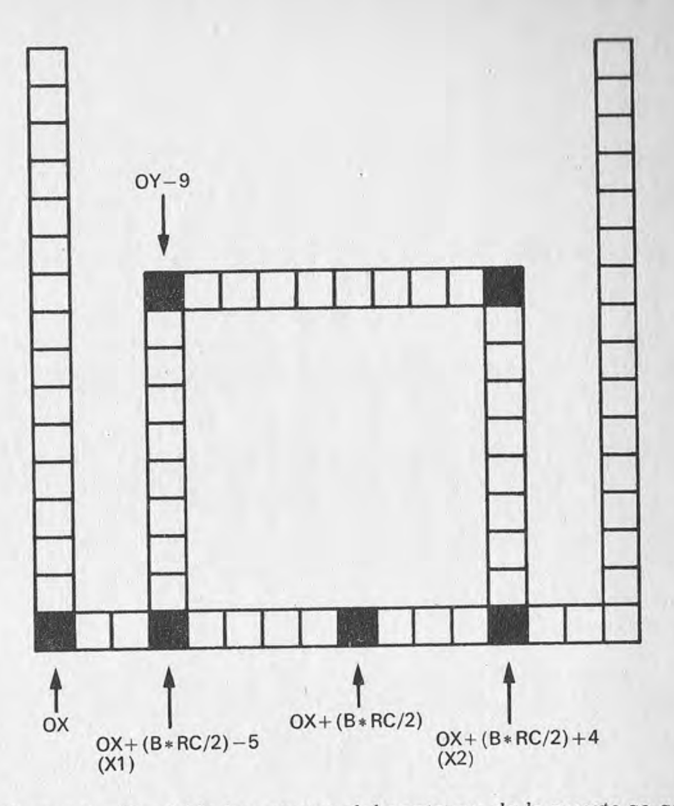


Figura 4.—También los puntos del contorno de la puerta se calculan en función de la posición del origen del rascacielos. Han sido definidos de tal modo que la puerta esté siempre centrada en relación a la base, cualquiera que sea su anchura.

tos "X1" y "X2" para, a continuación, trazar el contorno de la puerta en sí.

Pasamos seguidamente a las ventanas (líneas 660-730). Hay una fila vertical por cada RASC de anchura de la base (véase de nuevo la Fig. 2), desde el primer piso hasta arriba. En la figura 5 se muestra la posición de cada ventana, referida al "RASC cuadrado" en la que se halla. Esta posición es válida independientemente de que se encuentre de veras entre dos muros o rodeada por otras ventanas. La sección del programa que dibuja las ventanas comprende tres ciclos FOR...NEXT anidados uno dentro del otro; el más externo (línea 680) usa como índice la variable I9, que se desplaza en el sentido de la altura del rascacielos. Los límites del ciclo (desde 1 hasta H-1) son tales que las ventanas se dibujan a partir del primer piso hasta el último. El segundo bucle, con índi-

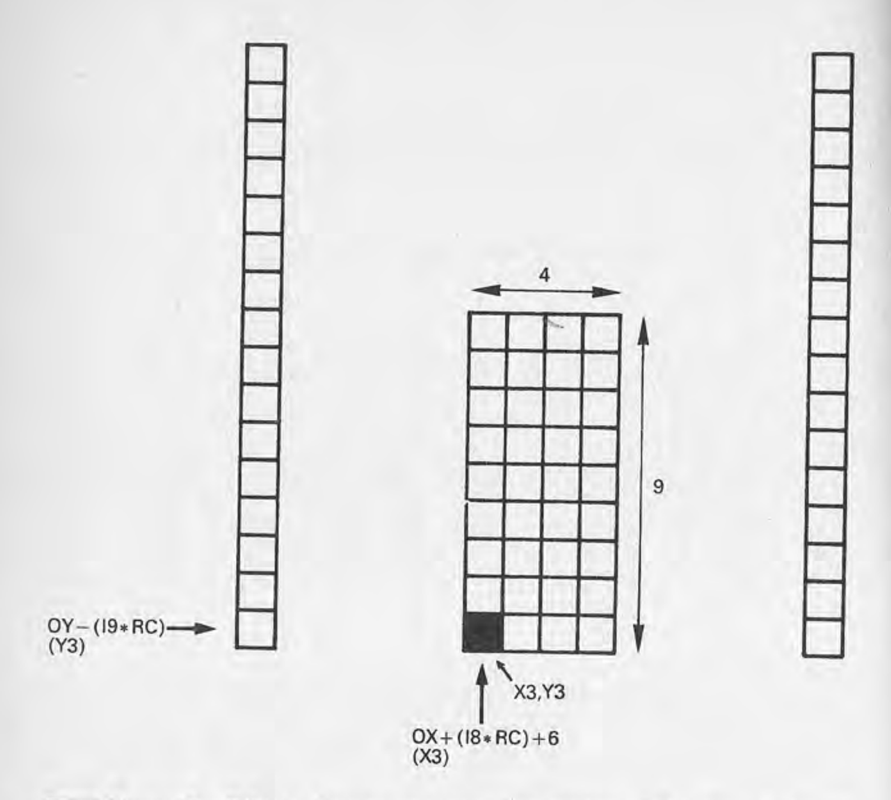


Figura 5.—Dibujo de una ventana. Los muros visibles en la figura pueden existir o no.

ce I8 (línea 680), se desplaza de izquierda a derecha paralelamente a la base: para cada valor asumido por I8 es dibujada una ventana.

Las ventanas son unos rectángulos llenos, cuya posición está en función de un punto de coordenadas "X3,Y3". Pueden ver que las dos expresiones que calculan "X3" e "Y3" (línea 690) contienen las coordenadas del origen y hacen al mismo tiempo referencia a los índices I8 e I9. De esta manera se cumple el principio general por el que cada elemento del rascacielos debe tener una posición definida con relación al origen. Una vez calculado el punto (X3, Y3) el tercer ciclo (línea 700), con variable I7, se encarga de dibujar la ventana, que sabemos que es un rectángulo con el vértice inferior izquierdo en la posición indicada por "X3" e "Y3". Sus dimensiones, 4×9 pixels, están determinadas tanto por los límites del ciclo en cuestión (de 0 a 3, es decir, cuatro veces) como por la constante 8 que aparece en la instrucción LINE en la línea 710.

El resultado global de los tres bucles vistos hasta ahora es el de dibujar todas las ventanas que se hallen en el cuerpo del rascacielos de abajo a arriba y (si la anchura es mayor que un RASC) de izquierda a derecha.

El siguiente paso es dibujar el saliente, que en el programa ha sido llamado "terminación" (líneas 750-810). Hay una novedad con relación a las secciones precedentes: mientras todos los rascacielos poseen puerta, muros y ventanas, la terminación es facultativa. Hay la posibilidad de escoger entre dibujarla o no por medio de la variable T, que es utilizada aquí como flag (un flag —literalmente "bandera"— es un indicador del tipo si/no). Si el flag T posee un valor distinto de cero; tal terminación será dibujada; de otro modo, el techo de la construcción será plano. Este es un ejemplo de cómo es posible variar con facilidad la figura alterando el valor de los parámetros que la describen, sin tener para ello que modificar las instrucciones del programa.

Existe además una condición que determina si se puede dibujar el saliente: el hecho de que el rascacielos posea una anchura superior a un RASC. En efecto, se ha definido que sólo aquellos cuya anchura es superior a uno puedan tener la terminación (línea 770). Ello se debe a que ésta posee una anchura igual a un RASC y se halla situada en el cento del techo. Este otro factor constituye una protección contra los planteamientos erróneos; es una buena norma proyectar un programa de manera que cualquier elección equivocada por parte del operador no produzca un mal funcionamiento o incluso el bloqueo del programa en sí.

El dibujo de la terminación se realiza de una manera que ya nos es familiar. En primer lugar se calculan las coordenadas (siempre en relación con el origen), en este caso X4, X5, Y4 e Y5 (línea 780). Después, sirviéndose de las mismas, una serie de instrucciones LINE dibujan los segmentos necesarios (líneas 790-810). Para comprender mejor las relaciones entre los puntos puede ver la Fig. 6.

También las ventanas de la terminación, más pequeñas que las anteriores y siempre emparejadas, son trazadas (líneas 830-890) con criterios parecidos a lo que ya vimos a propósito de las ventanas principales: para cada pareja de ventanas se calculan las posiciones de dos de sus puntos característicos (en este caso X6, Y6 y X7, en la línea 860); luego otro ciclo se encarga de dibujarlas en relación con esos puntos. El cálculo de X6, X7 e Y6 hace referencia a las coordenadas X4,Y4, que habían sido calculadas anteriormente para dibujar la terminación y que, a su vez, estaban en relación con el origen (OX, OY). De esta manera continúa el principio por el cual todos los elementos del rascacielos deben ser dibujados en relación con el origen.

El último elemento de la construcción es la punta (líneas

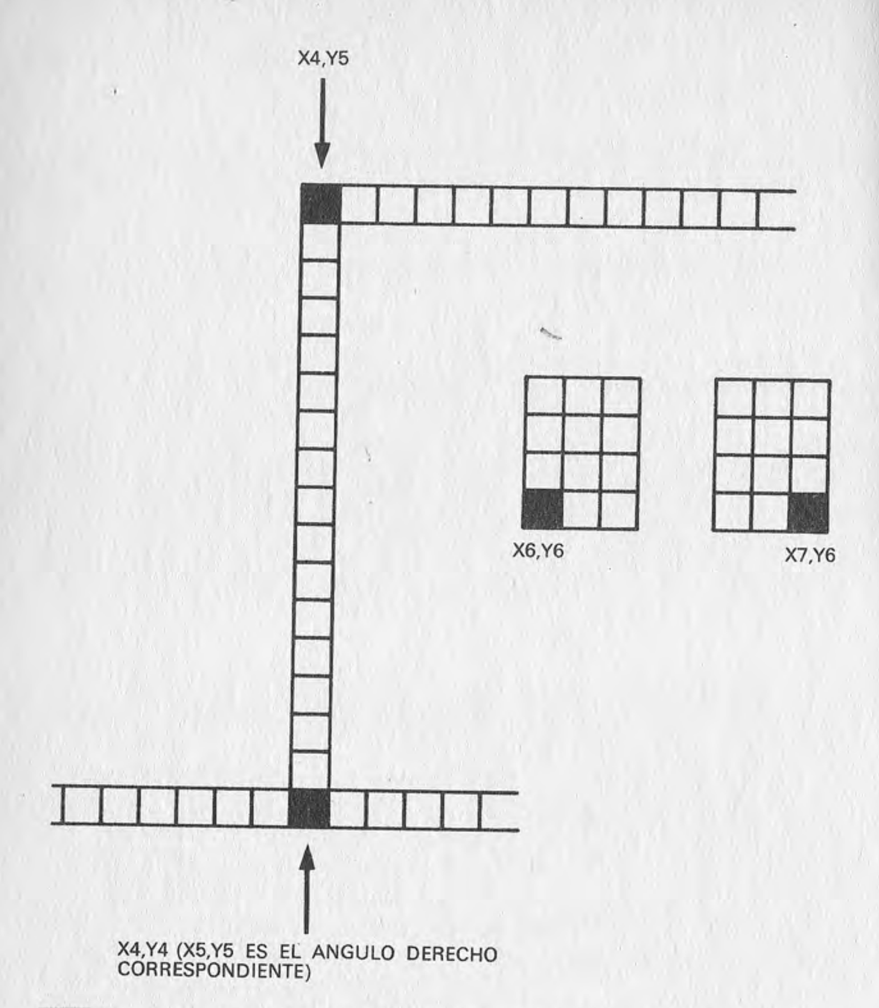


Figura 6.—Relación entre los principales puntos de la terminación y los de sus ventanas. Aquí se muestra tan sólo el extremo izquierdo de una terminación; su anchura depende de la del rascacielos.

910-970). Este también será opcional, dependiendo del valor del flag P (línea 930). Una punta está constituida por cuatro segmentos unidos como indica el esquema de la figura 7. Siguiendo el mismo procedimiento de siempre, también en este caso se calcula en primer lugar la posición de un punto característico (líneas 940-950), de coordenadas X8,Y8 (véase la figura). El cálculo incluye una corrección en la coordenada "Y" que entra en acción si el

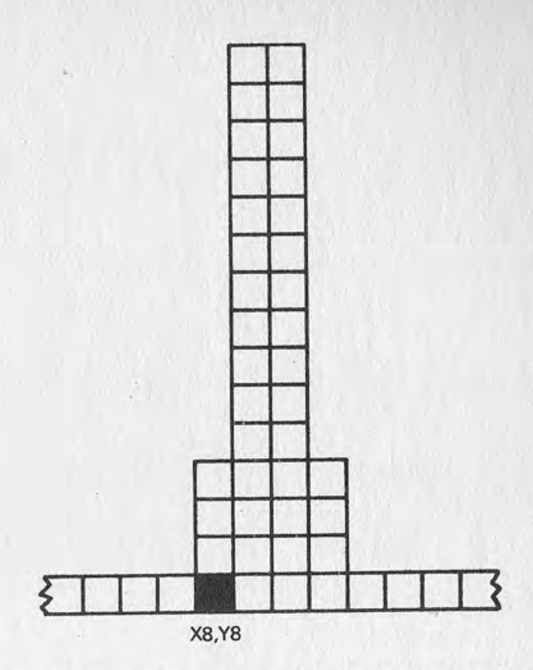


Figura 7.—Dimensiones y posición de una punta con relación al punto X8,Y8.

rascacielos posee también terminación. La corrección es efectuada por el test condicional que se halla en la línea 950 (nótese que aquí también actúa la protección anteriormente citada, que considera la existencia de una terminación en la cima del rascacielos sólo si éste es más ancho que un RASC).

Con esto hemos terminado el examen de la subrutina que efectúa el dibujo de todos los elementos del rascacielos. Como hemos visto, sus efectos son totalmente dependientes del valor de cuatro parámetros: B, H, T y P, y de las coordenadas del origen OX,OY.

Volviendo a observar el programa globalmente se nota que hay dos secciones distintas capaces de llamar a la subrutina de dibujo. La primera de ellas (líneas 140-200) interviene en el caso de que se quiera obtener la imagen del paisaje predefinido; la segunda, si se solicita uno casual (líneas 220-300).

Como habíamos anticipado, todo el paisaje visible en la figura 2 está descrito completamente por medio de un grupo de líneas DATA que se hallan reunidas al final del programa en una sección reservada para ellas (líneas 1020-1070). Estos datos pueden ser fácilmente alerados por el usuario del programa por si de-

sea darse cuenta mejor de su funcionamiento; es más, es conveniente tratar de hacerlo, sobre todo si no se es muy experto en gráficos computerizados. La facilidad con la que es posible modificar el comportamiento de un programa concebido de esta manera (es decir el paisaje que aparece en la pantalla) puede ser sorprendente y les animará, sin duda, a realizar por sí mismos algunos proyectos gráficos basándose en conceptos parecidos a los que han sido examinados en este ejemplo.

La organización de las líneas DATA es muy sencilla. La primera cantidad (línea 1040) representa el número de rascacielos que hay que dibujar, en este caso 12; las siguientes, tomadas de cuatro en cuatro, constituyen los parámetros que son pasados a la subrutina de dibujo para definir las características de cada uno de los rascacielos que debe aparecer en la pantalla. En el ejemplo del programa 2.5, que dibuja doce edificios, los parámetros que hay que pasar son en total 48. Su orden es el ya citado: B, H, T y P, correspondientes a base, altura, terminación (sí/no) y punta (sí/no). Al observar los primeros datos de la línea 1050 se nota inmediatamente que el primer rascacielos tendrá una base de dos RASC, una altura de seis, poseerá terminación pero no tendrá punta. La imagen que aparece en la pantalla confirma que eso es cierto.

La sección "paisaje predefinido" (líneas 140-200) lee en primer lugar el número de rascacielos NR en la línea 160, luego ejecuta un ciclo (líneas 170-200) que toma cuatro parámetros cada vez y llama a la susbrutina de dibujo para interpretarlos. Después de cada rascacielos, la coordenada "X" del origen es desplazada hacia la derecha un número de pixels igual a la anchura de la base (línea 190). El dibujo en pantalla se lleva a cabo, por tanto, de izquierda a derecha, ya que, como ha sido señalado otras veces, la posición de todo el rascacielos depende de la del origen (OX,OY).

¡Cuidado con un detalle!: No ha sido prevista ninguna protección contra una descripción del paisaje tal que lo haga salirse de los márgenes de la pantalla; esto puede constituir un tema interesante para un ejercicio.

Si, en cambio, lo que se desea es un paraje casual, entra en acción el segmento del programa siguiente (líneas 220-300), que genera aleatoriamente los valores de los cuatro parámetros B, H, T y P. La manera de utilizar la función RND nos es ya conocida; en este caso (línea 240) los valores posibles para B van de 1 a 3, y los de H, de 3 a 10 (para hacer una prueba, se puede intentar variar estos parámetros).

Seguramente les causará sorpresa el método utilizado para obtener "T" y "P" en las líneas 270 y 280. Esto se basa en el hecho de que "RND(1)>.5" constituye una expresión lógica a la que el BASIC asigna el valor cero si es falsa, o un valor distinto de cero en caso de ser verdadera. Como RND(1) es un número aleatorio comprendido entre 0 y 1, será mayor que 0.5 aproximadamente en el 50 por 100 de los casos y menor que 0.5 en el restante 50 por 100. El resultado es que la probabilidad de que un rascacielos tenga terminación es del 50 por 100, y lo mismo vale para la punta.

El dibujo del paisaje arbitrario termina cuando a la derecha de la pantalla no queda espacio suficiente para otro edificio (el control está en la línea 250). Este método es bastante inexacto, ya que aunque no haya espacio suficiente para el rascacielos calculado, podría ser bastante para un edificio con una base más pequeña. Se puede intentar, como ejercicio, modificar de forma adecuada el programa para que el paisaje termine siempre en la posición más a la derecha posible de la pantalla (dibujando un rascacielos de dimensiones adecuadas).

También en el caso del paisaje casual después del dibujo de cada edificio el origen es desplazado hacia la derecha lo suficiente como para trazar el siguiente (línea 290).

CAPITULO IV

DIBUJO DE FIGURAS GEOMÉTRICAS

Problemas de escala: cuadrados rectangulares y círculos elípticos

Cuando se trabaja con programas, a la hora de hacer figuras geométricas surge inmediatamente la cuestión de las proporciones de la pantalla. El problema reside en esto: si un programa concebido para trazar un cuadrado (como el 2.3) dibuja en cambio un rectángulo, o vemos aparecer una elipse cuando lo que queríamos era una circunferencia, significa que la escala del eje "X" es distinta de la del eje "Y". En otras palabras, un mismo número de puntos forma un segmento de una longitud, digamos, de diez centímetros sobre el eje "X", mientras que sobre el eje "Y" traza otro de medida diferente. Toda imagen de la pantalla tendrá las proporciones alteradas, como si hubiera sido "estirada" en una dirección y aparecerá distorsionada.

Cambiando el punto de vista podemos decir que esto ocurre cuando los pixels no son cuadrados sino rectangulares. El inconveniente puede producirse debido a dos causas: o el ordenador genera una página gráfica con los ejes a escalas diferentes, o el monitor (o televisor) no está correctamente regulado, de tal modo que distorsiona la imagen. El Apple posee una resolución de 280×192 puntos; el Commodore 64, de 320×200; ninguno de los dos corresponde exactamente a la relación 4/3 que caracteriza las dimensiones de una pantalla. Por tanto, a pesar de utilizar un monitor perfectamente regulado, se tendrá en ambos casos una distorsión. El Commodore 64 se aleja de una manera particular de la relación exacta.

```
10 REM *** PROGRAMA 2.6 ***
20 REM
30 REM TEST ESCALA
40 REM
50 REM
60 HIRES 1,0
70 REM
80 LINE 0,0,199,0,1
90 LINE 199,0,199,199,1
100 LINE 199,199,0,199,1
110 LINE 0,199,0,0,1
120 REM
130 GET T$:IF T$="" THEN 130
READY
```

El programa 2.6 permite comprobar si el inconveniente se presenta y, en caso afirmativo, en qué medida. Puede hacerse trazando lo que debería ser un cuadrado: si la figura que aparece no es ésa nos hallaremos frente a un descalibramiento de escala; habrá entonces que medir con una regla los lados del rectángulo que aparece en pantalla.

Definimos como "factor de escala" la relación entre la medida del lado vertical "Y" y la del horizontal "X". Esta relación será mayor o menor que uno, según que predominen la dimensión "Y" o la "X". Una vez conocido el factor corrector adecuado puede normalizarse la situación procediendo como indicamos en el programa 2.7: todas las coordenadas del eje "X" son multiplicadas por el factor de escala. En este programa habrá que rectificar, naturalmente, la línea 60 e introducir la verdadera relación que hemos obtenido (el valor 1.2 sirve tan sólo como ejemplo). De ahora en adelante todos los programas que traten de figuras geométricas incluirán esta corrección; está claro que habrá que tomar cada vez la precaución de introducir el valor correcto del factor de escala, indicado siempre con la variable FE.

```
10 REM *** PROGRAMA 2.7 ***
20 REM
30 REM CORRECCION DE ESCALA
40 REM
50 REM
60 FE=1.2:REM FACTOR DE ESCALA
70 HIRES 1,0
80 REM
```



```
90 LINE 0*FE,0,199*FE,0,1
100 LINE 199*FE,0,199*FE,199,1
110 LINE 199*FE,199,0*FE,199,1
120 LINE 0*FE,199,0*FE,0,1
130 REM
140 GET T$:IF T$="" THEN 140
READY
```

El recorte de las figuras (clipping)

Los programas examinados hasta ahora han sido desarrollados para producir imágenes más o menos determinadas desde un principio, pero al ir adentrándonos en los gráficos computerizados a menudo se halla uno frente al dibujo y manipulación de figuras de naturaleza variable.

Cuando el programa está destinado a un uso general hay que tener en cuenta un problema: puede ocurrir que algunos segmentos de la imagen tengan una medida y/o orientación tal que se "salgan" de la pantalla. Con el Commodore 64 esto ocurre si se indica una coordenada "X" menor que cero o mayor que 319, o una coordenada "Y" menor que cero o mayor que 199. Cuando se da este caso, el Commodore 64 lo considera (correctamente) un error, produciendo la interrupción inmediata del programa en cuestión.

Este inconveniente puede resolverse si incluimos en los programas que lo requieran una subrutina de "clipping" (recorte) como la que aparece en el programa 2.8. El objetivo de una subrutina de este tipo es el examinar las coordenadas de cada segmento que hay que trazar en pantalla para intervenir luego adecuadamente en caso de que uno de sus extremos se halle fuera de los límites de la página gráfica. La intervención consiste en cortar el segmento de manera que quede interrumpido en correspondencia con el borde de la pantalla.

```
10 REM *** PROGRAMA 2.8 ***
20 REM
30 REM CLIPPING DE SEGMENTOS
40 REM
50 REM
60 XI=0:XD=319:YA=0:YB=199
70 HIRES 1,0:GOTO 320
80 REM
90 REM CLASIFICACION DE PUNTOS
100 REM
```

```
110 I(I)=X(I)<XI
120 D(I)=X(I)<XD
130 A(I)=Y(I)>YA
140 B(I)=Y(I)>YB
150 RETURN
160 REM
170 REM CALCULA LOS EXTREMOS
180 REM
190 GOSUB 110
200 V=0:IF I(1)*I(2)+D(1)*D(2)+A(1)*A(2)+
    B(1)*B(2)<>0 THEN 280
210 I=1:IF I(1)+D(1)+A(1)+B(1)<>0 THEN 240
220 I=2:IF I(2)+D(2)+A(2)+B(2)<>0 THEN 240
230 V=1:GOTO 280
240 IF I(I)<>0 THEN Y(I)=Y(1)+(XI-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(I)=XI:GOTO 190
250 IF D(I)<>0 THEN Y(I)=Y(1)+(XD-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(I)=XD:GOTO 190
260 IF A(I)<>0 THEN X(I)=X(1)+(YA-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(I)=YA:GOTO 190
270 IF B(I)<>0 THEN X(I)=X(1)+(YB-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(I)=YB:GOTO 190
280 RETURN
290 REM
300 REM PROGRAMA PRINCIPAL
310 REM
320 LINE XI,YA,XD,YA,1
330 LINE XD,YA,XD,YB,1
340 LINE XD,YB,XI,YB,1
350 LINE XI,YB,XI,YA,1
360 READ NP:FOR J=1 TO NP:FOR I=1 TO 2
370 READ X(I),Y(I):GOSUB 110
380 NEXT I
390 GOSUB 200:IF V<>0 THEN LINE X(1),Y(1),X(2),Y(2),1
400 NEXT J
410 GET T$:IF T$="" THEN 410
420 END
430 REM
440 REM EXTREMOS DE LOS SEGMENTOS
450 REM
460 DATA 5:REM No.SEGMENTOS
```



```
470 DATA -20,10,350,50
480 DATA 30,130,150,60
490 DATA -30,-10,50,210
500 DATA 30,250,340,50
510 DATA 210,-30,340,120
READY
```

Vamos a echar un vistazo a este procedimiento. En primer lugar son definidas cuatro variables (línea 60) destinadas a contener el valor máximo permitido para las coordenadas "X" e "Y". Se trata de XI (X izquierda), XD (X derecha), YA (Y arriba) e YB (Y abajo). A continuación sigue el verdadero proceso de clipping, dividido en dos subrutinas diferentes (líneas 90-150 y 170-280). El programa principal (300-420) lee los extremos de los segmentos que hay que dibujar de un grupo de líneas DATA para después someterlas al clipping. En este caso se trata de cinco segmentos (líneas 440-510); el resultado de la operación será posteriormente trazado en la pantalla.

En primer lugar, el programa principal dibuja los límites de la página gráfica (líneas 320-350). Esto no es esencial para la ejecución del procedimiento de clipping, pero sirve para mostrar mejor el efecto. Después lee el número de líneas que hay que dibujar; valor que es utilizado como límite del ciclo que lee las coordenadas de los extremos de todos los segmentos. Para cada uno de ellos (variable J) es ejecutada dos veces la línea 370. Su función es la de leer las coordenadas de los dos extremos de un solo segmento de las líneas DATA; luego las enviará una por una a la subrutina de clasificación de los puntos (90-150). Esta, por tanto, recibe las cuatro coordenadas de los dos extremos, contenidas en las variables X(1), Y(1) y X(2), Y(2). Cada uno de estos puntos es clasificado en base al esquema que aparece en la figura 1. Sustancialmente, se determina si el punto en cuestión está dentro o no de la página gráfica, y si no lo está, en qué otra zona se halla.

Queremos recordar también que el valor numérico resultante de una expresión lógica como X(I)<XI (líneas 110-140) está determinado por el hecho de que ésta sea verdadera o falsa, y le corresponde, respectivamente, el valor 1 (en algunos ordenadores -1) ó 0. Por tanto, tras haber llamado por dos veces a la subrutina, los dos extremos del segmento estarán clasificados por el contenido de las variables I(1), D(1), A(1), B(1) para el primer punto, e I(2), D(2), A(2), B(2) para el segundo.

Posteriormente, la línea 390 llama a la segunda subrutina que, según los resultados obtenidos en la primera, calcula los extremos del segmento correspondientes tan sólo a la parte "visible" del total. En la misma línea (390) se procede también al dibujo del

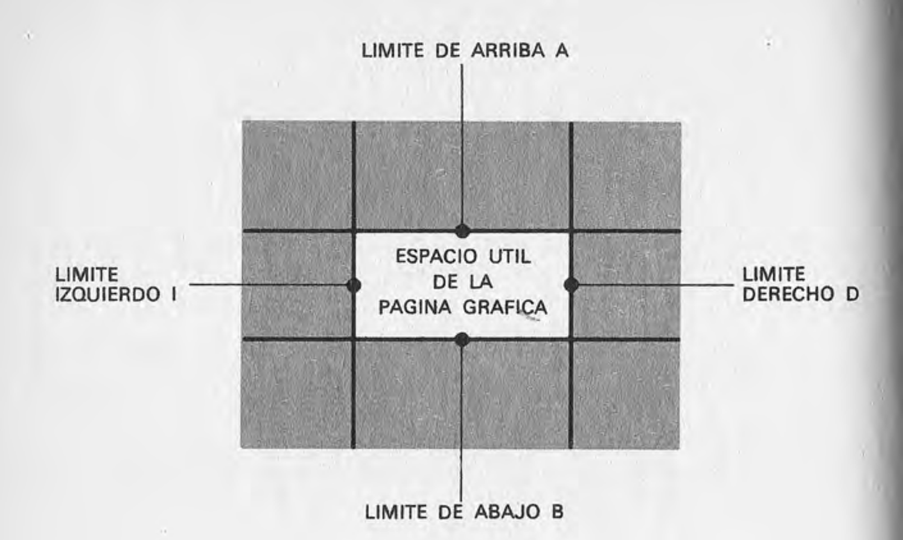


Figura 1.—La subrutina de clasificación de los puntos determina si cada uno de los extremos de un segmento se halla o no fuera de uno de los cuatro límites de la zona visible.

mismo, pero sólo si la variable V lo clasifica como "visible" (tomando en tal caso un valor distinto de cero). Este examen es hecho en la línea 200:

```
200 V=0: IF I(1)*I(2)+D(1)*D(2)+A(1)*A(2)+B(1)*B(2)<>0 THEN280
```

Si el segmento en cuestión posee ambos extremos situados más allá de un mismo límite, uno de los productos, por ejemplo D(1)*D(2), resultará distinto de cero. Si esto ocurre, el proceso de clipping termina aquí, señalando por medio de la variable V que el segmento examinado no posee parte alguna dentro de la página gráfica y que, por tanto, es "invisible".

Las líneas 210 y 220 comprueban si el segmento tiene alguno de los dos extremos situados fuera de los límites:

```
210 I=1:IF I(1)+D(1)+A(1)+B(1)<>0 THEN240
220 I=2:IF I(2)+D(2)+A(2)+B(2)< >0 THEN240
```

Si no ocurre esto, es innecesario el clipping para ese segmento y será declarado en seguida visible (en la línea 230) ya que se halla por completo dentro del espacio permitido. En caso contrario habrá que calcular un nuevo extremo (líneas 240-270) situado en la

intersección entre el segmento y el límite conflictivo (el método matemático utilizado para este fin se describe más abajo).

La figura 2 muestra el caso de varios segmentos: uno invisible, otro completamente visible y un tercero que lo está sólo en parte y que tendrá por tanto que someterse al proceso de clipping. Los extremos A y B de este último (véase figura) son transformados por el clipping en otros dos, indicados como C y D que, naturalmente, se hallarán en alguno de los límites, de manera que una instrucción LINE trazará tan sólo la parte del segmento originario que resulta visible, sin provocar errores.

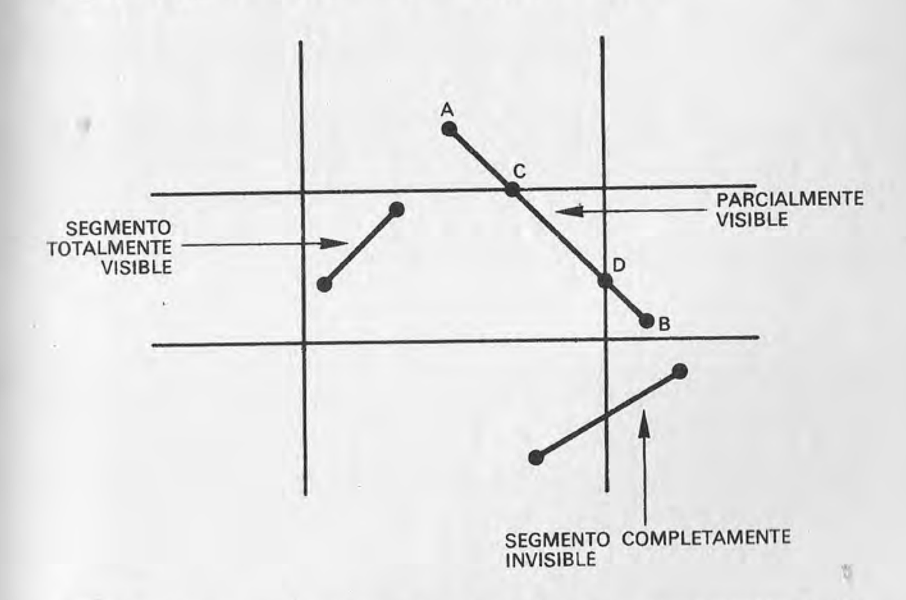


Figura 2.—Los verdaderos extremos A y B de un segmento parcialmente visible son sustituidos en el clipping por otros dos, C y D, situados en los límites de la zona útil.

Para entender a fondo el procedimiento utilizado para calcular los nuevos extremos de los segmentos sometidos a un clipping hay que estar un poco familiarizados con la geometría analítica. Haciendo referencia a la Figura 3, aquellas personas que tengan cierta base en esta materia reconocerán con facilidad la ecuación del segmento AB, que es:

Y=Y1+(X-X1)(Y2-Y1)/(X2-X1)

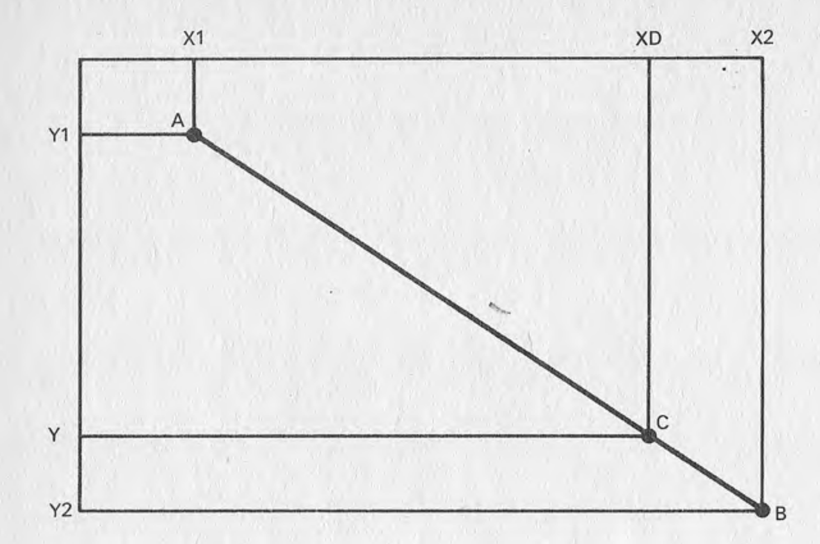


Figura 3.—Un segmento de extremos A(X1,Y1) y B(X2,Y2) es sometido a clipping. El extremo "B" se sustituye por "C", cuyas coordenadas "XD" e "Y" se obtienen mediante las ecuaciones descritas en el texto.

Si se quiere hallar la intersección con el extremo derecho bastará introducir en la ecuación la coordenada horizontal del límite que nos interesa, en este caso XD, en lugar del genérico "X". El punto de intersección tendrá, por tanto, de coordenadas:

X=XD
Y=Y1+(XD-X1)(Y2-Y1)/(X2-X1)

Teniendo en cuenta que en el programa 2.8 los extremos están contenidos en un vector, esta ecuación equivale a la que aparece en la línea 250:

```
IF D(I)<>0 THEN Y(I)=Y(1)+(XD-X(1))*
(Y(2)-Y(1))/(X(2)-X(1)):X(I)=XD: GOTO190
```

Las intersecciones con los otros tres bordes se calculan de la misma manera (líneas 240-270).

Una vez concluido su trabajo el programa espera a que sea pulsada una tecla cualquiera (línea 410), de manera que se pueda ver fácilmente el resultado. Este mismo procedimiento se adoptará en muchos de los próximos programas.

Todos los polígonos que queramos

Los programas siguientes servirán para que nos familiaricemos con algunos de los procedimientos utilizados para obtener figuras geométricas. El primero de ellos (2.9) dibuja polígonos regulares con un número de lados variable a partir de 3.

```
10 REM *** PROGRAMA 2.9 ***
20 REM
30 REM POLIGONOS
40 REM
50 REM
60 CX=160:CY=100:FE=1.1
70 GOTO 240
80 REM
90 REM TRAZADO DE EJES
100 REM
110 FOR X=0 TO CX*2-1 STEP 2
120 PLOT X,CY,1:NEXT X
130 FOR Y=0 TO CY*2-1 STEP 2
140 PLOT CX,Y,1:NEXT Y
150 RETURN
160 REM
170 REM ERROR
180 REM
190 PRINT "QSEAMOS SERIOS!"
200 RETURN
210 REM
220 REM PROGRAMA PRINCIPAL
230 REM
240 PRINT "♥QQ"
250 INPUT "¿CUANTOS LADOS?";L
260 IF L<3 THEN GOSUB 190:GOTO 250
270 INPUT "QDISTANCIA DE LOS VERTICES
    AL CENTRO";D
280 IF D>CY THEN GOSUB 190:GOTO 270
290 HIRES 1,0:GOSUB 110:A=2*PI/L
300 FOR I=0 TO L
310 X=CX-(D*SIN(I*A)+.5)*FE
320 Y=CY-D*COS(I*A)+.5
330 IF I=0 THEN 350
```

```
340 LINE XP,YP,X,Y,1
350 XP=X:YP=Y:NEXT I
360 REM
370 GET T$:IF T$="" THEN 370
380 NRM:PRINT "♥QQ"
390 INPUT " ¿CONTINUO? (<RETURN>=SI)";R$
400 IF R$="" THEN 240
410 PRINT '♥"
READY
```

Tras la petición del número de lados del polígono que va a ser dibujado, el verdadero trabajo comienza con el trazado de dos ejes cartesianos en el centro de la pantalla (líneas 110-150). Los ejes son discontinuos (un punto sí y otro no) para que no se confundan con la figura que será dibujada más tarde. El procedimiento es el siguiente: un segmento imaginario que tiene un extremo en el centro de los ejes (es decir, en el nuevo origen), se hace girar en sentido antihorario de manera que el otro extremo del mismo localice uno por uno todos los vértices del polígono (Fig. 4). Al unir posteriormente estos puntos se obtendnrá el dibujo deseado.

El cálculo de la posición del segundo extremo del segmento rotatorio utiliza las nociones de seno y coseno, ilustradas en la figura 5a. Es importante recordar que en BASIC los ángulos son expresados en radianes en lugar de los corrientes grados sexagesimales; con estas unidades el ángulo de una vuelta completa no vale 360 grados, sino dos veces pi (π) (aproximadamente 6,28) radianes (Fig. 5b). Este es el motivo por el cual, para calcular el ángulo de rotación correspondiente a cada vértice, aparece en la línea 290 la expresión A=2*PI/L. Vamos a refrescar un poco la memoria acerca de los radianes:

- 1/2 pi radianes equivalen a un ángulo de 90 grados;
- pi radianes equivalen a 180 grados;
- 2 pi radianes constituyen una vuelta completa, es decir 360 grados;
- un radián corresponde a 57.29577 grados;
- en general, un ángulo de G grados corresponde a (G/180)*pi radianes;
- por consiguiente R radianes corresponden a (R/pi)*180 grados sexagesimales.

Para dibujar el polígono se hace uso de un ciclo FOR...NEXT que se inicia en la línea 300. Aprovechando las relaciones trigonométricas citadas arriba, para cada iteración del ciclo (líneas

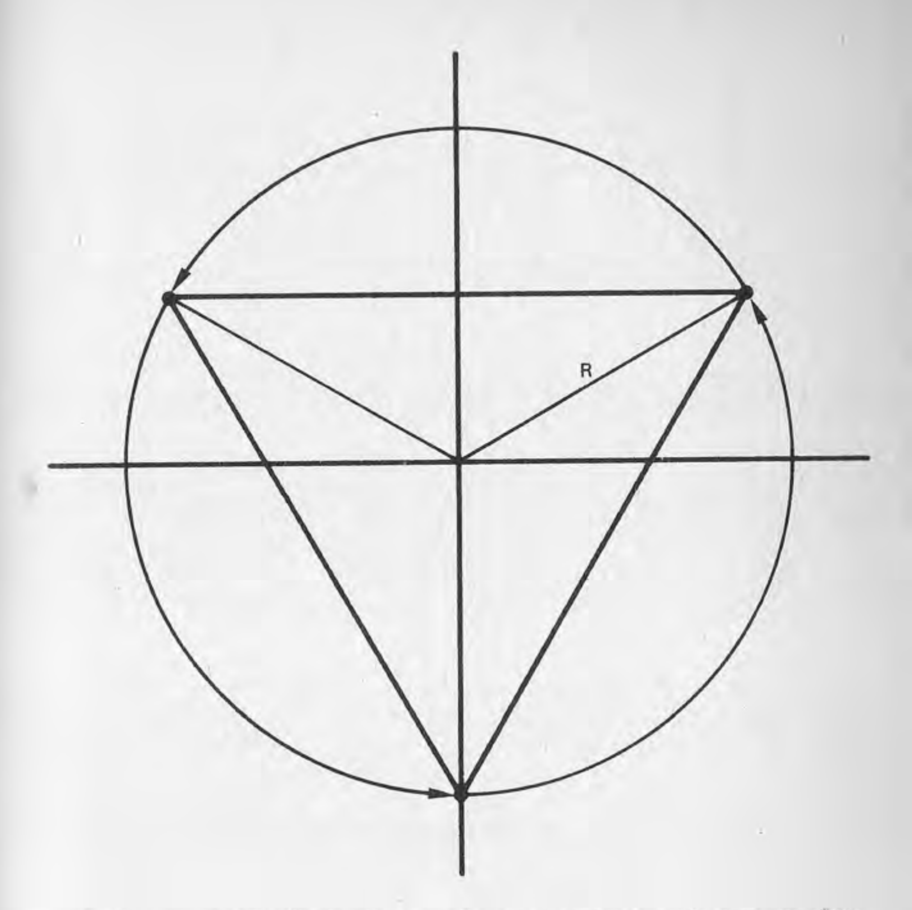
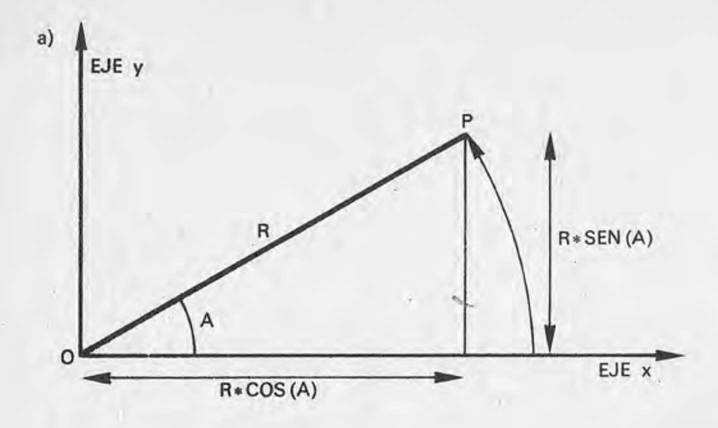


Figura 4.—El dibujo de un polígono regular, en este caso un triángulo equilátero, se efectúa haciendo girar alrededor del origen un segmento de longitud dada un número de veces igual al de los vértices. Uniendo luego los puntos así hallados se obtendrá el polígono.

310-320) se calcula la posición de un vértice, que es unido al anterior con una línea recta (exceptuando, naturalmente, el primero que no tiene ninguno anterior). Nótese que los ejes cartesianos que hemos dibujado en la pantalla no se corresponden con los utilizados habitualmente para trabajar en la página gráfica, de modo que es necesario transformar las coordenadas que se han obtenido en otras referidas al mismo sistema de referencia de siempre, de forma que el origen se encuentra en el ángulo superior izquierdo de la pantalla. La transformación se produce también en las lí-



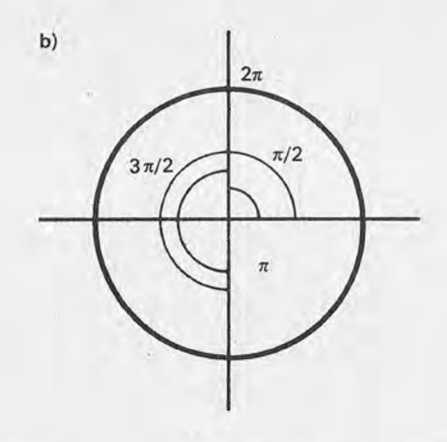


*Figura 5.—a) Un segmento R gira en sentido contrario a las manecillas del reloj un ángulo "A" hasta hallarse en posición OP. Para cada valor de "A" la proyección de OP sobre el eje "X" es igual a R*COS(A) y sobre el eje "Y" R*SIN(A). Evidentemente, estas cantidades representan también las coordenadas cartesianas "X" e "Y" del punto "P" para un ángulo "A". b) Valores en radianes de distintos ángulos.*

neas 310 y 320, al incluir en el cálculo las coordenadas del centro de la página gráfica CX y CY.

Tal vez les sorprenda la presencia en esas líneas del factor 0.5, que aparece tanto en el cálculo de "X" como en el de "Y". Los valores que se obtienen del cálculo de "X" e "Y" son fraccionarios, así que es conveniente redondearlos al entero más cercano antes de utilizarlos para localizar un pixel en la página gráfica. Ya hemos visto (programa 1.3) que la instrucción LINE considera tan

sólo la parte entera de las coordenadas, de modo que podemos utilizarla para obtener el redondeo automáticamente. Bastará con añadir antes la constante 0.5: en efecto, si el valor está comprendido entre 2 y 2.49999999, al añadir 0.5 se mantendrá menor que 3, por lo que su parte entera valdrá siempre 2 (que corresponde con el entero más cercano). Si, en cambio, estamos entre 2.5 y 2.99999999, al añadir 0.5 obtendremos siempre una cantidad mayor o igual a 3 (el entero más cercano en este caso). Considerando sólo la parte entera se obtiene precisamente 3. Aplicaremos siempre esta técnica en los siguientes programas.

Hay otra observación en relación con el factor de escala (FE) cuyo valor, si es distinto de uno, hace que una línea a 45 grados no se corresponda con una diagonal de pixel. El fenómeno se observa fácilmente dando el valor 1 al factor de escala y dibujando un polígono de cuatro lados; luego se le asigna un valor diferente (por ejemplo, el correcto) y se repite el dibujo del cuadrado.

El programa 2.9 permite dibujar nuevos polígonos sin tener que dar nuevamente al RUN. Basta con responder adecuadamente a la pregunta formulada en la línea 390, apretando la tecla de RETURN. Nótese que ha sido también prevista una protección contra posibles errores de planteamiento (subrutina 190 y controles en las líneas 260 y 280).

Dibujar un círculo.
Primer método: con la raíz cuadrada

Existen muchos métodos para dibujar un círculo en la pantalla de un ordenador personal. El hecho de estudiar varios de ellos servirá para mostrar cómo se puede resolver un mismo problema de diferentes maneras. El programa 2.10 logra este fin sin hacer uso de la trigonometría, valiéndose sólo de la operación "raíz cuadrada". En efecto, para cualquier punto de una circunferencia que tenga como centro el origen de coordenadas (Fig. 6) vale la relación:

$$R^2=X^2+Y^2$$

donde "R" es el radio del círculo, y "X" e "Y" las coordenadas del punto elegido. Esto no es otra cosa que el conocido teorema de Pitágoras: su utilización está justificada porque el triángulo OPH es rectángulo (uno de sus ángulos es de 90 grados).

```
10 REM *** PROGRAMA 2.10 ***
20 REM
30 REM TRAZAR UN CIRCULO  (1)
```

```
40 REM
50 REM
60 HIRES 1,0:CX=160:CY=100:R=80:
   P=32:FE=1.1
70 DEF FNY(X)=SQR(R*R-X*X)
80 REM
90 REM TRAZADO DE LOS EJES
100 REM
110 FOR X=0 TO CX*2-1 STEP 2:PLOT X,CY,1:
    NEXT X
120 FOR Y=0 TO CY*2-1 STEP 2:PLOT CX,Y,1:
    NEXT Y
130 REM
140 REM PROGRAMA PRINCIPAL
150 REM
160 S=2*R/P
170 FOR X1=R TO -R STEP -S
180 Y=FNY(X1)+CY+.5:X=X1*FE+CX+.5
190 IF X1=R THEN 210
200 LINE XP,YP,X,Y,1
210 XP=X:YP=Y:NEXT X1
220 FOR X1=-R TO R STEP S
230 Y=-FNY(X1)+CY+.5:X=X1*FE+CX+.5
240 LINE XP,YP,X,Y,1
250 XP=X:YP=Y:NEXT X1
260 REM
270 GET t$:IF T$="" THEN 270
READY
```

Al inicio del programa, además de las variables "CX" y "CY" es definida la función FNY(X), con el fin de determinar la coordenada "Y" a partir de la "X" y del valor del radio "R". La coordenada "X" se hace variar en el interior de dos ciclos FOR...NEXT situados en las líneas 170-210 y 220-250; se hallará cada vez el valor "Y" correspondiente. Los puntos localizados de esta manera son unidos por medio de segmentos por las instrucciones LINE; el círculo obtenido está así formado en realidad por una serie de segmentos rectilíneos, pero su número es suficiente como para que la figura resulte bastante aproximada.

Como ocurre en el programa 2.9 también aquí es necesario intervenir en los valores calculados para que el círculo se halle situado en el centro de la pantalla. Esto se hace simplemente com-

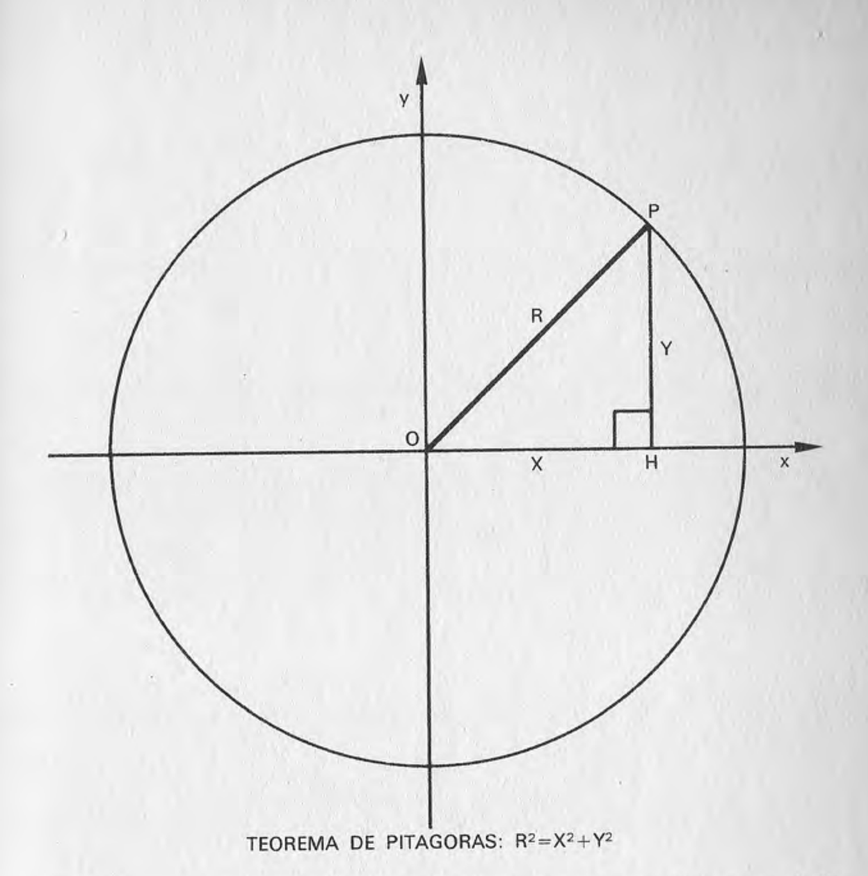


Figura 6.—Para dibujar un círculo podemos utilizar el teorema de Pitágoras, que proporciona una relación entre el radio y las coordenadas de un punto cualquiera de la circunferencia (indicado con "P").

binando cada valor de "X" e "Y" con las coordenadas del centro del monitor CX y CY (línea 180).

Decíamos que este resultado era satisfactorio, pero al observar con atención la pantalla notaremos que el dibujo no se forma con segmentos iguales; la razón estriba en que los valores de "Y" son calculados al desplazar la coordenada "X" de manera uniforme, de forma que los puntos resultantes están más separados en las cercanías del eje "X" (Fig. 7). Este fallo resulta particularmente evidente si se disminuye el número de segmentos que constituyen la circunferencia; para ello bastará con reducir el valor de "P" de 32 a 10 (línea 60). El resultado se parecerá aún menos que antes a un círculo.

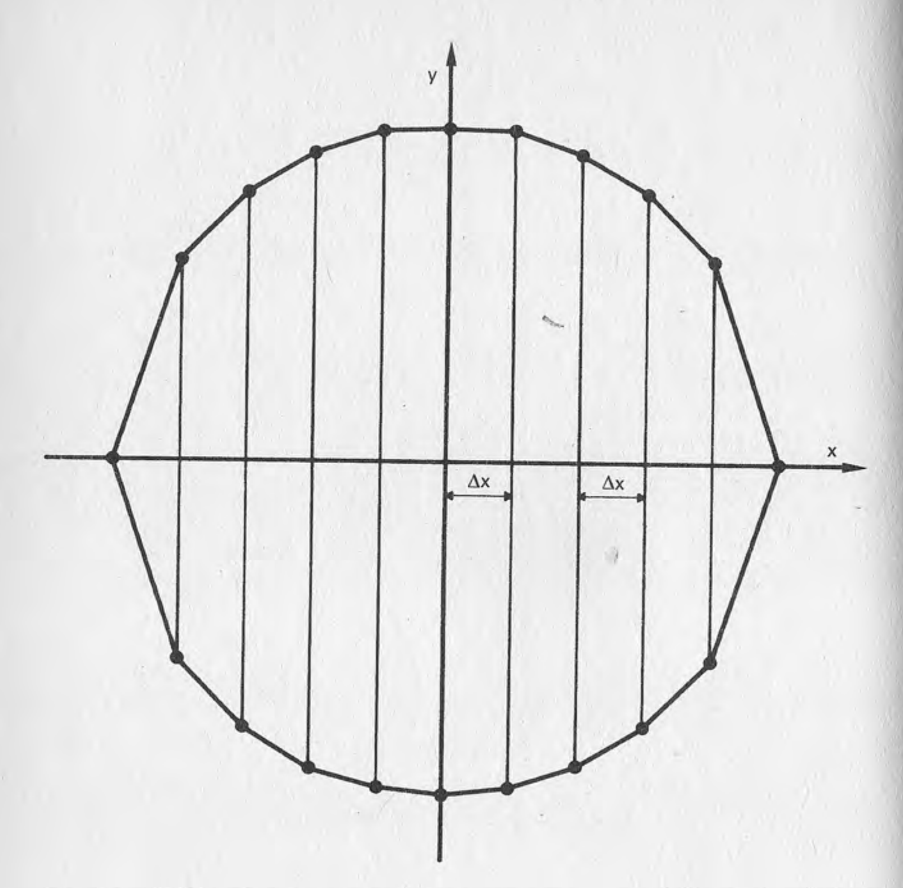


Figura 7.—El defecto de la técnica de la figura 6 resulta particularmente evidente si se disminuye el número de puntos que hay que calcular; así aparece con claridad la distorsión en las proximidades del eje "X".

Segundo método: seno y coseno

En este caso se utilizan los conceptos de trigonometría ya vistos a propósito de los polígonos (programa 2.9). En realidad el método es prácticamente el mismo, pues para nuestros propósitos un círculo equivale a un polígono regular con un número de lados lo bastante grande como para obtener un grado de aproximación adecuado (en este caso N=32). Para convencerse es suficiente con coger el programa 2.9 y solicitar que sea dibujado un polígono de 30 o más lados: se obtendrá prácticamente un círculo.


```
10 REM *** PROGRAMA 2.11 ***
20 REM
30 REM TRAZAR UN CIRCULO (2)
40 REM
50 REM
60 HIRES 1,0
70 CX=160:CY=100:FE=1.1:R=80:N=32:
   DA=2*PI/N
80 REM
90 REM DIBUJO DE LOS EJES
100 REM
110 FOR X=0 TO 2*CX-1 STEP 2:PLOT X,CY,1
    :NEXT X
120 FOR Y=0 TO 2*CY-1 STEP 2:PLOT CX,Y,1
    :NEXT Y
130 REM
140 REM PROGRAMA PRINCIPAL
150 REM
160 FOR I=0 TO 2*PI STEP DA
170 X=R*COS(I)*FE+CX+.5
180 Y=R*SIN(I)+CY+.5
190 IF I=0 THEN 210
200 LINE XP,YP,X,Y,1
210 XP=X:YP=Y:NEXT I
220 REM
230 GET T$:IF T$="" THEN 230
READY
```

Con esta técnica no se producirán nunca distancias irregulares entre los distintos puntos calculados, ya que lo que se divide en partes iguales durante el trazado no es el eje "X", sino el círculo. En efecto, lo que cambia durante el cálculo es el ángulo entre el radio y el eje "X" (variable I), que en cada ciclo aumenta en un valor DA (línea 160) hasta completar la vuelta completa ($2*\pi$ radianes); de ahí que sea $DA=2*\pi/N$ (línea 70).

Tercer método: puntos correlativos con el precedente

Un método interesante, con el que se obtiene el mismo resultado que en el anterior, hace uso de un algoritmo más "astuto" y veloz: cada punto es localizado a partir de las coordenadas del precedente, de modo que no se tienen que calcular cada vez el seno y el coseno.

```
10 REM *PROGRAMA 2.12 **
20 REM
30 REM TRAZAR UN CIRCULO (3)
40 REM
50 REM
60 HIRES 1,0
70 CX=160:CY=100:FE=1.1:X1=80:Y1=0
80 N=32:DA=2*PI/N:SI=SIN(DA):CO=COS(DA)
90 REM
100 REM TRAZADO DE LOS EJES
110 REM
120 FOR X=0 TO 2*CX-1 STEP 2:PLOT X,CY,1:NEXT X
130 FOR Y=0 TO 2*CY-1 STEP 2:PLOT CX,Y,1:NEXT Y
140 REM
150 REM PROGRAMA PRINCIPAL
160 REM
170 XP=X1*FE+CX+.5:YP=CY
180 FOR I=DA TO 2*PI STAP DA
190 P=X1*CO-Y1*S1:Y1=Y1*CO+X1*SI:X1=P
200 X=X1*FE+CX+.5:Y=CY-Y1
210 LINE XP,YP,X,Y,1
220 XP=X:YP=Y:NEXT
230 REM
240 GET T$:IF T$="" THEN 240
READY
```

Supongamos haber hallado ya las coordenadas X1, Y1 de un punto P1, correspondiente a un ángulo "A" sobre la circunferencia de radio "R". Ya sabemos que:

X1 = R*COS(A)
Y1 = R*SIN(A)

Sea ahora un punto P2 situado también sobre la circunferencia, pero sobre un radio que forma un ángulo igual al de P1, es decir, "A", aumentado en un valor DA. Las coordenadas X2,Y2 del nuevo punto P2 estarán dadas por:

X2 = R*COS(A+DA)
Y2 = R*SIN(A+DA)

Dado que hay una relación trigonométrica que establece para nuestro caso que:

R*COS(A+DA) = R*COS(A)*COS(DA)—R*SIN(A)*SIN(DA)
R*SIN(A+DA) = R*SIN(A)*COS(DA)+R*COS(A)*SIN(DA)

si sustituimos en estas expresiones las coordenadas de los puntos P1 y P2 obtendremos:

X2 = X1*COS(DA)—Y1*SIN(DA)
Y2 = Y1*COS(DA)+X1*SIN(DA)

Pero el ángulo DA es constante durante todo el programa, ya que representa el incremento angular utilizado para poder pasar de un punto cualquiera al siguiente (en efecto, se obtiene en la línea 80 al dividir el ángulo completo por el número de puntos que hay que obtener). Por consiguiente también SIN(DA) y COS(DA) son cantidades constantes y por tanto se pueden calcular de una vez para siempre al inicio del proceso (línea 80, denominadas SI y CO). De aquí se deriva que tras haber fijado la posición del primer punto (en la línea 70), para calcular el siguiente pueden utilizarse las relaciones vistas anteriormente, que no requieren operación trigonométrica alguna, sino sólo algunas sumas, restas y multiplicaciones.

Cuando hayamos obtenido el segundo punto utilizaremos la misma fórmula para calcular el tercero (a partir esta vez del segundo) y todos los demás hasta completar el círculo. Como en los otros casos, al unir los puntos entre sí con segmentos se obtendrá el círculo deseado.

Para los tres métodos es válida la consideración de que cuanto mayor sea el número de puntos calculados más pequeños serán los segmentos y, por tanto, mejor la calidad del dibujo, que se parecerá cada vez más a una verdadera circunferencia.

La elipse

Esta figura, que se obtiene si "estiramos" un círculo, será dibujada por el programa 2.13.

```
10 REM *** PROGRAMA 2.13 ***
20 REM
30 REM TRAZAR UNA ELIPSE
40 REM
50 REM
60 HIRES 1,0:CX=160:CY=100
70 FE=1:A=100:B=60:N=32:DA:2*PI/N
80 REM
```

```
90 REM TRAZADO DE LOS EJES
100 REM
110 FOR X=0 TO CX*2-1 STEP 2:PLOT X,CY,1
    NEXT X
120 FOR Y=0 TO CY*2-1 STEP 2:PLOT CX,Y,1
    NEXT Y
130 REM
140 REM PROGRAMA PRINCIPAL
150 REM
160 FOR AL=0 TO 2*PI STEP DA
170 X1=A*COS(AL):Y1=B*SIN(AL)
180 X=CX+FE*X1+.5:Y=CY-Y1+.5
190 IF AL=0 THEN 210
200 LINE XP,YP,X,Y,1
210 XP=X:YP=Y:NEXT AL
220 REM
230 GET T$:IF T$="" THEN 230
READY
```

El procedimiento es casi idéntico al seguido para trazar un círculo con el segundo método (2.11); la única diferencia estriba en que en el cálculo aparecen las variables "A" y "B" en lugar del radio "R" (línea 170). Estos dos parámetros corresponden, respectivamente, al semieje mayor y al semieje menor de la elipse, como muestra la figura 8.

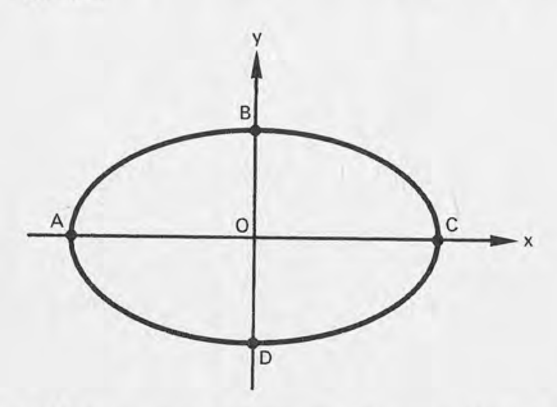


Figura 8.—Una elipse con centro en el origen de coordenadas. El semieje mayor $\overline{AO}$ (=$\overline{OC}$) corresponde al parámetro A utilizado en el cálculo, mientras que el semieje menor $\overline{BO}$ (=$\overline{OD}$) corresponde al parámetro B.

Las ecuaciones utilizadas describen una elipse que tiene su centro en el origen de coordenadas, de modo que será necesario (como en el caso de la circunferencia) transformar los valores obtenidos para trasladar la figura al centro de la página gráfica, donde está situado el origen de los nuevos ejes cartesianos dibujados por el programa.

Curvas en polares

Consideremos ahora una circunferencia cuyo radio varía constantemente durante el trazado, en función del ángulo en el que se halla en ese instante. Según la regla utilizada precisamente para definir el radio, se obtendrán distintos tipos de curva, que tienen en común el hecho de estar centradas en el origen de los ejes. Esto es lo que hace el programa 2.14. Para no crear confusión con las figuras resultantes, esta vez no hemos trazado los ejes, sino sólo los bordes de la pantalla.

Vamos a aprovechar esta ocasión para observar cómo están definidos los puntos de una curva de este tipo. Ya hemos comentado anteriormente que, en un plano cartesiano, cada punto se define por sus dos coordenadas, "X" e "Y", llamadas también coordenadas rectangulares. Pero también hay otra manera de identificar un punto "P": esta vez los dos parámetros son la distancia "R" al origen y la medida del ángulo comprendido entre el segmento que une el origen con "P" y el eje "X" (Fig. 9). Estas coordenadas se llaman coordenadas polares.

Es posible pasar de coordenadas polares a rectangulares sirviéndose de dos sencillas relaciones que ya hemos mencionado:

X=R*COS(A)
Y=R*SIN(A)

Naturalmente, el programa adopta estas relaciones para actuar en la forma corriente en la página gráfica (en donde pueden utilizarse fácilmente sólo las coordenadas rectangulares). Recuerde que en BASIC los ángulos son expresados siempre en radianes.

```
10 REM *** PROGRAMA 2.14 ***
20 REM
30 REM CURVAS EN POLARES
40 REM
50 REM
60 HIRES 1,0:CX=160:CY=100:FE=1.1:
   N=64:AF=2*PI:DA=AF/N
```

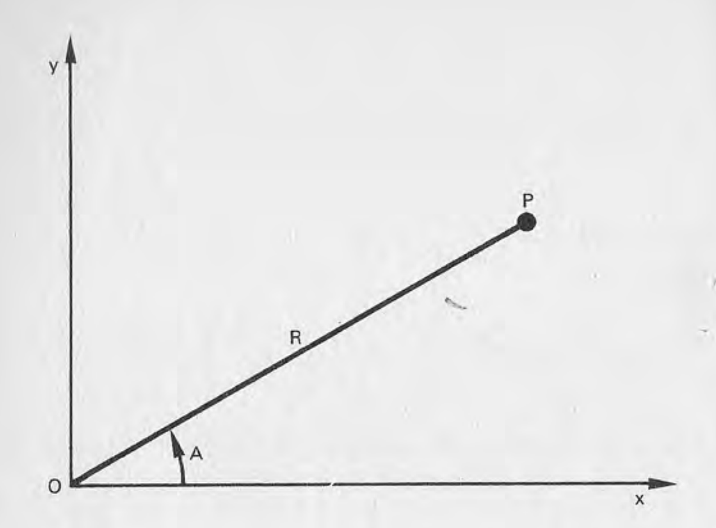


Figura 9.—Un punto "P" en un plano cartesiano puede estar definido por medio de dos coordenadas polares. La primera es la distancia "R" entre el punto y el origen; la otra es el ángulo "A" comprendido entre el eje "X" y el segmento $\overline{OP}$, medido en sentido antihorario.

```
70 DEF FNR(A):50-40*SIN(3*A)
80 REM
90 LINE 0,0,319,0,1
100 LINE 319,0,319,199,1
110 LINE 319,199,0,199,1
120 LINE 0,199,0,0,1
130 REM
140 FOR A=0 TO AF+.01 STEP DA
150 R=FNR(A)
160 X1=R*COS(A):Y1=R*SIN(A)
170 X=CX+FE*X1+.5:Y=CY-Y1+.5
180 IF X>319 OR X<0 OR Y>199 OR Y<0
    THEN F=0:GOTO 220
190 IF F=0 THEN F=1:GOTO 210
200 LINE XP,YP,X,Y,1
210 XP=X:YP=Y
220 NEXT A
230 REM
240 GET T$:IF T$="" THEN 230
READY
```


En el programa 2.14 la función que hace variar el radio según el ángulo está definida en la línea 70 como FNR(A). Resulta interesante probar varias de estas funciones y observar los resultados. Puede experimentar con los siguientes, modificando también el valor de N y de AF (ángulo final) como está indicado (Pi indica la constante π):

* FN R(A) = 2*A (definir N = 64 y AF = 8*PI)
* FN R(A) = 80*SIN(A) (N = 64; AF = PI)
* FN R(A) = 80*COS(2*A) (N = 64; AF = 2*PI)
* FN R(A) = 80 (N = 64; AF = 2*PI)
* FN R(A) = 70+10*COS(6*A) (N = 64; AF = 2*PI)
* FN R(A) = 70+10*COS(12*A) (N = 128; AF = 2*PI)
* FN R(A) = 60+30*COS(6*A) (N = 128; AF = 2*PI)
* FN R(A) = 60+30*RND(1) (N = 64; AF = 2*PI)
* FN R(A) = 80*(COS(A)—0.5) (N = 64; AF = 2*PI)
* FN R(A) = 40*(COS(2*A)—1) (N = 64; AF = 2*PI)
* FN R(A) = 50*(COS(2*A)—0.5) (N = 64; AF = 2*PI)
* FN R(A) = 80*COS(3*A) (N = 64; AF = PI)
* FN R(A) = 2*EXP(A/6.28) (N = 128; AF = 8*PI)
* FN R(A) = 70—40*SIN(2*A) (N = 64; AF = 2*PI)
* FN R(A) = 50—40*SIN(3*A) (N = 64; AF = 2*PI)
* FN R(A) = 50—40*SIN(A*A) (N = 128; AF = 2*PI)

Como se ve, las hay para todos los gustos. Algunas funciones dan resultados que nos son muy familiares. Dos de ellas dibujan una circunferencia. La figura 10 muestra los resultados que aparecen en la pantalla al experimentar con algunas de las funciones propuestas.

Obsérvese que el límite del bucle de cálculo que se inicia en la línea 140 no es AF, sino AF+0.01. Esto es necesario para evitar que, debido a problemas de aproximación del ordenador al efectuar las sumas (es decir en cada NEXT A) el bucle termine un ciclo antes de lo previsto. Cuidado: con otros ordenadores distintos del Commodore 64 no tiene por qué producirse tal inconveniente.

Como última nota debemos indicar que el programa contiene una protección a fin de que una figura no pueda salirse de las imágenes de la pantalla y genere así un error (línea 180). Se trata de una precaución indispensable en un programa que, como éste, se presta a experimentar diferentes funciones.

Figura 10.—He aquí algunos ejemplos de curvas en polares. Las que aparecen en la figura han sido obtenidas con el programa 2.14, tras introducir cuatro ecuaciones diferentes:
*a) FNR(A) = 50-40*SIN(3*A)* *c) FNR(A) = 70+10*COS(6*A)*
*b) FNR(A) = 2*EXP(A/6.28)* *d) FNR(A) = 50-40*SIN(4*A)*

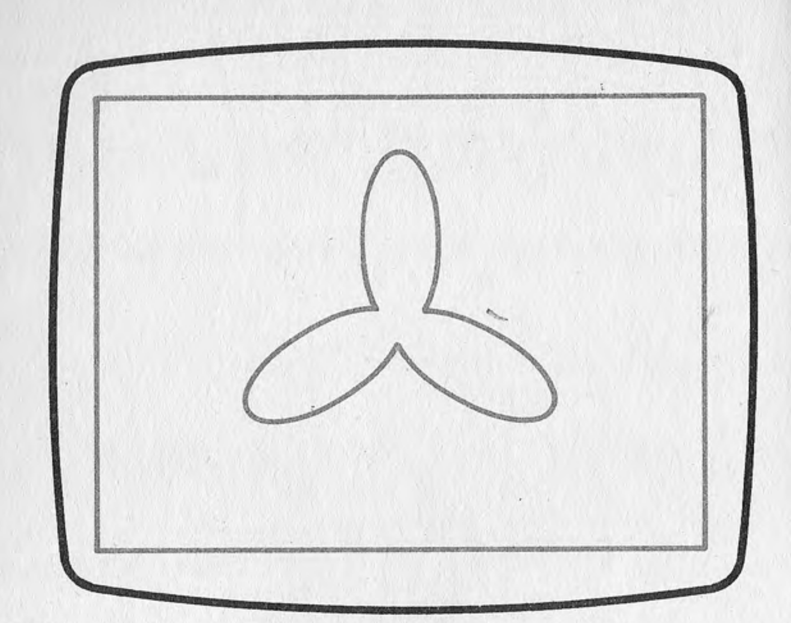


Figura 10a.—FNR(A) = 50-40*SIN(3*A).

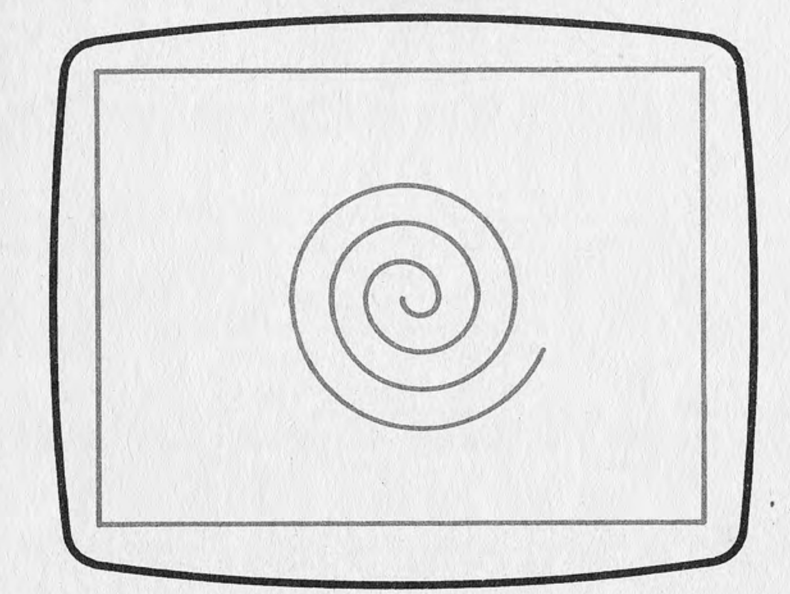


Figura 10b.—FNR(A) = 2*EXP(A/6.28).

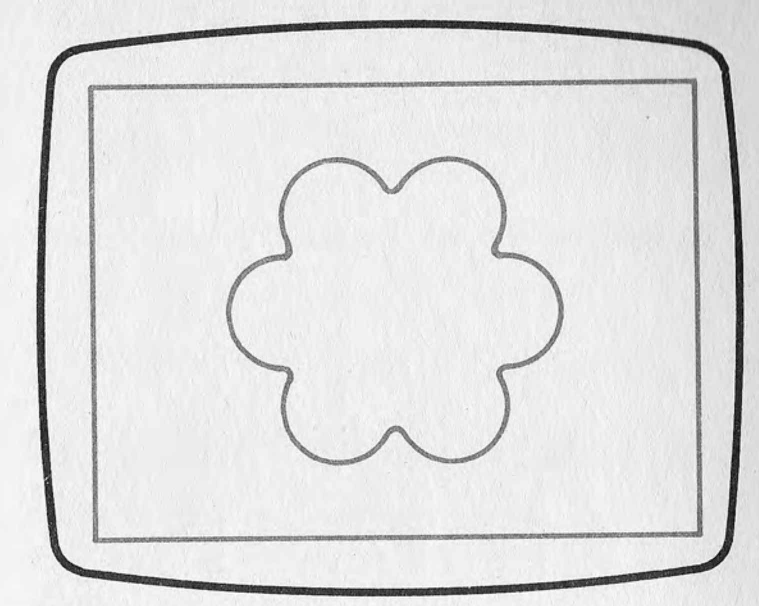


Figura 10c.—FNR(1) = 70+10*COS(6*A).

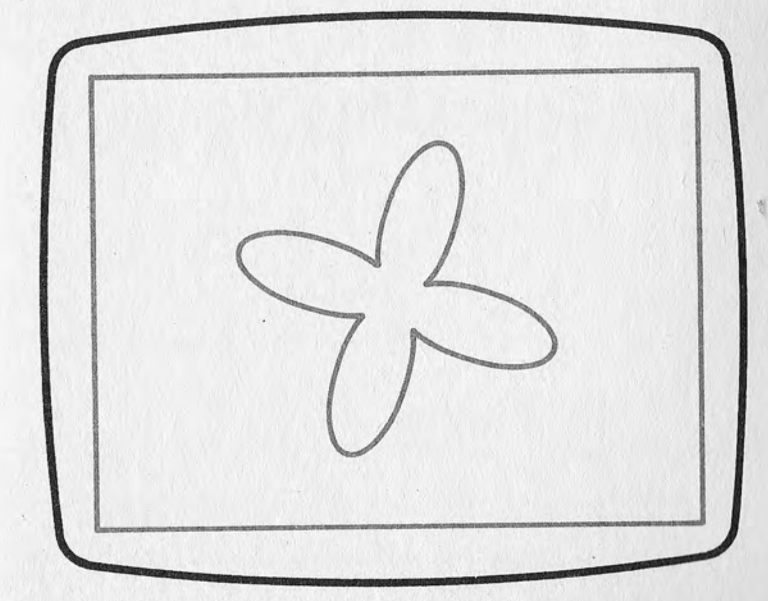
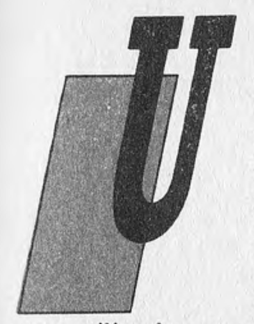


Figura 10d.—FNR(1) = 50-40*SIN(4*A).

CAPITULO V

MANIPULACIÓN DE FIGURAS EN DOS DIMENSIONES

Utilización de las matrices en los gráficos



na magnífica cualidad de los gráficos computerizados es la de poder manipular a placer una imagen. Las técnicas más sofisticadas capacitan al operador para modelar completamente los objetos que aparecen en pantalla de tal forma que puede proyectar nuevas formas o estructuras sin tener que recurrir a un modelo real. Y, aún más, es posible incluso generar imágenes con un parecido a la realidad tal que son utilizadas en películas, como es el caso ya citado de TRON.

Nosotros nos ceñiremos a experimentar algunas de las técnicas de transformación de imágenes bidimensionales, pues con un ordenador personal o doméstico tenemos ciertas limitaciones. Ciertamente no estaremos a la altura de los resultados citados anteriormente, pero la técnica que se expone en el programa 2.15 es suficiente para una interesante serie de realizaciones.

```
10 REM *** PROGRAMA 2.15 ***
20 REM
30 REM TRANSFORMACIONES
40 REM
50 REM
60 CX=160:CY=100:FE=1.1
70 REM
80 PRINT "♥":GOTO 180
90 REM
100 REM TRAZADO DE LOS EJES
```

```
110 REM
120 FOR X=0 TO CX*2-1 STEP 2:PLOT X,CY,1
    :NEXT X
130 FOR Y=0 TO CY*2-1 STEP 2:PLOT CX,Y,1
    :NEXT Y
140 RETURN
150 REM
160 REM PROGRAMA PRINCIPAL
170 REM
180 PRINT "[S] ESTE PROGRAMA MUESTRA EL EFECTO"
190 PRINT "DE UNA MATRIZ DE TRANSFORMACION"
200 PRINT "DE 2X2 ELEMENTOS."
210 PRINT "[Q]LAS COORDENADAS DE LA FIGURA ESTAN"
220 PRINT "INTRODUCIDAS EN EL PROGRAMA EN FORMA"
230 PRINT "DE LINEAS 'DATA'"
240 PRINT "[Q][Q] INTRODUCE LA MATRIZ: [Q]"
250 INPUT "A=";A:INPUT "B=";B
260 INPUT "C=";C:INPUT "D=";D
270 REM
280 HIRES 1,0:GOSUB 120
290 RESTORE:READ NP
300 FOR P=1 TO NP:READ X1,Y1
310 XT=A*X1+C*Y1:YT=B*X1+D*Y1
320 X=CX+FE*XT+.5:Y=CY-YT+.5
330 IF P=1 THEN 350
340 LINE XP,YP,X,Y,1
350 XP=X:YP=Y:NEXT P
360 REM
370 GET T$:IF T$="" THEN 370
380 NRM:PRINT "[Q][Q] <ESPACIO>=BASTA,<RETURN>=OTRA VEZ";
390 GET T$
400 IF T$=CHR$(32) THEN PRINT "[Q][Q] ":END
410 IF T$=CHR$(13) THEN 180
420 GOTO 390
430 REM
440 REM PUNTOS DE LA FIGURA
450 REM
460 DATA 5:REM No. PUNTOS
470 DATA 0,0,40,0,40,40,0,40,0,0
READY
```


Partimos de un cuadrado que tiene el vértice inferior izquierdo en el origen y que es definido, como en otros programas, por medio de una serie de DATAs en las líneas 460 y 470. El programa solicita al operador cuatro parámetros: A, B, C y D (líneas 180-260), después lee uno por uno los puntos de la figura que debe dibujar y los somete al proceso de transformación (línea 310). Para cada uno de los puntos las coordenadas transformadas (XT,YT) son posteriormente "ajustadas" en la manera ya conocida, con el fin de desplazar el origen de los ejes al centro de la pantalla (línea 320) y utilizadas más tarde para trazar la figura.

Pero, ¿cuál es el mecanismo de la transformación? Para cada pareja (X,Y), que representa un punto, se efectúa su multiplicación por una matriz compuesta por los parámetros A, B, C y D, obteniendo así un nuevo punto "transformado". Matemáticamente todo ello se expresa de la siguiente manera:

$$(X,Y)\cdot\begin{vmatrix} A & B \\ C & D \end{vmatrix} = (AX+CY,\ BX+DY) = (XT,\ YT)$$

Se observa, por tanto, que la multiplicación de las coordenadas de un punto (X,Y) por una matriz de cuatro elementos puede hacerse utilizando las consabidas operaciones de producto y suma.

Aquellas personas que no tienen muy claro qué es una matriz no vayan a asustarse: a grandes rasgos, y para lo que nos interesa, basta saber que se trata de un grupo de números dispuestos en rectángulo y delimitados por dos paréntesis, como es el caso de:

matriz de 2×3 de 2×2 de 2×2 de 1×1 de 2×1

$$\begin{vmatrix} 3 & 4 & 1 \\ 7 & 8 & 2 \end{vmatrix},\quad \begin{vmatrix} 7 & 0 \\ -2 & 4 \end{vmatrix},\quad \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \text{ y también } |3\quad 5| \text{ o } \begin{vmatrix} 8 \\ 3 \end{vmatrix}$$

Nótese que también (X,Y) es una matriz, compuesta en este caso por una sola línea (se les denomina entonces "vectores"). Podemos multiplicar matrices entre sí. Por el momento no vamos a examinar la manera general de hacerlo; baste saber que para que una matriz de a×b y otra de c×d se puedan multiplicar, existe una única condición: que "b" y "c" sean iguales. El resultado será entonces una matriz de a×d. Así vimos antes que al multiplicar la de 1×2 por la de 2×2 resultaba otra de 1×2. La primera de estas matrices representa en nuestro caso las coordenadas del punto (X,Y),

la otra se llama "matriz de transformación". Dando valores apropiados a los cuatro parámetros de esta matriz, es decir A, B, C y D, podremos obtener las transformaciones deseadas.

Existe una matriz que no produce ningún efecto, y se llama, por tanto, "matriz identidad". Se trata de

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

y tiene la propiedad de dejar inalterados todos los puntos. (Para convencerse basta que haga la prueba con la regla vista anteriormente.)

Podremos obtener algo un poco más interesante con una matriz del tipo:

$$\begin{vmatrix} A & 0 \\ 0 & D \end{vmatrix}$$

cuya función es la de efectuar una transformación de escala en la figura. La coordenada "X" de cada punto será multiplicada por el factor "A" y la "Y" será multiplicada por "D". Si "A" y "D" son iguales, la figura variará sus dimensiones manteniendo las mismas proporciones, de otro modo quedará deformada. En ese caso un cuadrado, por ejemplo, se transformará en rectángulo.

Otra posibilidad es ofrecida por la reflexión de la imagen con relación al eje "X" o al eje "Y" (Fig. 1). Estas transformaciones se obtienen utilizando, respectivamente, las dos matrices

$$\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix} \text{ y } \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}$$

Una vez más el motivo del por qué ocurre esto debería estar bastante claro si se tienen en cuenta los razonamientos anteriores. Basta que realice un ejemplo concreto para que se dé cuenta.

Una matriz asimétrica, por ejemplo

$$\begin{vmatrix} 1 & 1.2 \\ 0 & 1 \end{vmatrix}$$

producirá un "estiramiento" de la imagen, en este caso en dirección del eje Y. Efectivamente, si por ejemplo tenemos el punto

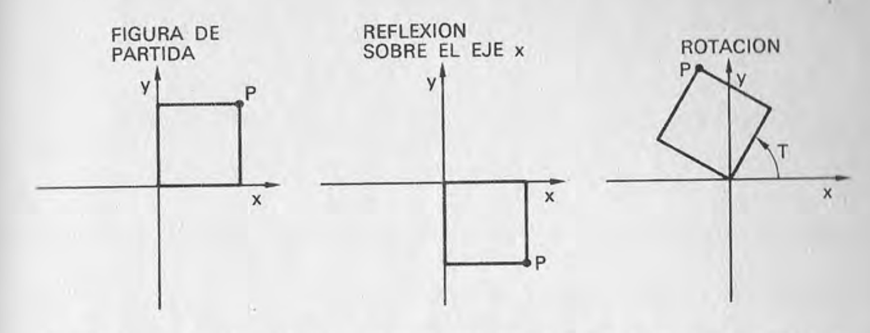


Figura 1.—Al aplicar a cada punto de la figura una matriz de transformación adecuada pueden obtenerse resultados como los de la figura. En este caso se trata de una reflexión sobre el eje "X" y una rotación alrededor del origen.

(3,3) al aplicarle la matriz de transformación anterior pasará a ser el (3, 6'6)

Otro efecto parecido, pero esta vez en el sentido del eje "X", puede obtenerse al intercambiar de lugar los elementos B y C.

Rotación alrededor del origen

Volvemos a tratar una vez más de trigonometría; la matriz:

$$\begin{vmatrix} COS(T) & SIN(T) \\ -SIN(T) & COS(T) \end{vmatrix}$$

tiene la propiedad de rotar toda la figura un ángulo T en sentido contrario a las agujas del reloj (recuérdese que aquí los ángulos se miden en radianes). La rotación resultante se hallará centrada en el origen (Fig. 1); en otras palabras, el punto (0,0) no se desplaza nunca de su posición. Para probar el efecto sobre el cuadrado del programa 2.15 basta con asignar a la matriz de transformación los cuatro valores siguientes, que corresponden a una rotación de 30 grados (es decir casi 0.523 radianes):

A=COS(T)=0.866
B=SIN(T)=0.5
C=-SIN(T)=-0.5
D=COS(T)=0.866

Con esta técnica, al calcular adecuadamente la matriz, podremos hacer rotar todas las figuras un ángulo cualquiera. Teniendo en cuenta que es posible modificar las líneas DATA (460-470) para codificar una figura diferente, por compleja que sea, empezará a hacerse una idea de lo potentes que pueden ser (y sobre todo con un uso muy general) los instrumentos puestos a disposición de los gráficos computerizados para elaborar imágenes.

Transformaciones sucesivas

Supongamos que queremos aplicar a la figura dos transformaciones, una tras otra; por ejemplo, una reflexión y luego una rotación. Para hacerlo tendríamos que utilizar, en primer lugar, una matriz de reflexión (que llamaremos M1) y, posteriormente, otra de rotación (M2).

Aparte el hecho de que el programa 2.15 no ha sido previsto para utilizar varias matrices en secuencia, existe otro método mejor. En lugar de aplicar a la figura primero M1 y luego M2 para obtener la transformación deseada, puede obtenerse el mismo resultado utilizando sólo una matriz M3 igual al producto de M1 por M2. Podremos así combinar anticipadamente cualquier transformación y obtener una sola matriz que, una vez aplicada a la figura, dará el mismo resultado que una secuencia de transformaciones individuales.

La regla para multiplicar entre sí dos matrices de 2×2 elementos, extensión de la vista anteriormente, es:

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} \cdot \begin{vmatrix} E & F \\ G & H \end{vmatrix} = \begin{vmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{vmatrix}$$

Pero ¡cuidado!: no es lo mismo multiplicar M1 por M2 que M2 por M1. Por regla general existe la tendencia a considerar siempre cierta la propiedad conmutativa del producto, pero ésta *no es válida* en el caso del producto de matrices.

Es fácil convencerse al observar que, geométricamente, al reflejar una figura sobre el eje "X" y rotarla luego un cierto ángulo se produce un efecto distinto del obtenido efectuando primero la rotación (del mismo ángulo) y luego la reflexión. Así, es natural que al hacer lo mismo matemáticamente el resultado sea igual.

Un ejercicio útil consiste en calcular una matriz de transformación equivalente a dos (o más) transformaciones sencillas en secuencia, y aplicarla luego a la figura del programa 2.15. Compárese entonces la imagen obtenida con la que habíamos esperado ver si hubiese sido posible aplicar las transformaciones una después de la otra: si los cálculos son correctos el resultado debería ser idéntico.

Un paso adelante: las coordenadas homogéneas

Todas las transformaciones precedentes poseen un aspecto en común: no afectan para nada al punto situado en el origen. Ninguna matriz de entre las examinadas es capaz de efectuar una traslación de toda la figura; en efecto, hemos visto que la multiplicación de un punto por una matriz de 2×2 elementos equivale al producto de las coordenadas del punto por algunas constantes (A, B, C y D), seguido de la suma de los cuatro resultados tomados dos a dos, pero en ningún caso se le suma una constante a las coordenadas.

Un instrumento de manipulación gráfica más potente nace al adoptar el uso de las llamadas coordenadas homogéneas. En ellas cada punto del plano puede ser localizado con tres números en lugar de con los dos de las coordenadas cartesianas o polares que ya conocemos. Diremos, por tanto, que:

- para definir un punto se utiliza una terna de valores (P,Q,R);
- esta terna es elegida de tal modo que las relaciones P/R y Q/R correspondan a las coordenadas cartesianas "X" e "Y";
- de aquí se deriva que, por cada punto específico (X,Y), existen infinitas formas de representarlo con las coordenadas homogéneas. Hy, efectivamente, infinitas combinaciones de P y R tales que X=P/R e Y=Q/R. Tomemos como ejemplo el punto (3,5); en coordenadas homogéneas corresponde tanto a (6,10,2) como a (9,15,3), (3,5,1) o cualquier otra terna que cumpla la relación X=P/R a Y=Q/R;
- entre las infinitas ternas permitidas utilizaremos para nuestros fines sólo la forma (P,Q,1) que recibe el nombre de *coordenada homogénea normalizada*. Considerando las relaciones anteriores esto equivale a escribir (X,Y,1), donde "X" e "Y" son las coordenadas cartesianas habituales del punto (X,Y).

En definitiva, respecto de la situación anterior ha habido pocos cambios. Entonces, ¿para qué todo este lío? Lo importante es que ahora cada punto está definido por tres números, así que podemos transformarlo utilizando matrices de 3×3 elementos. Resulta imposible multiplicar una matriz del tipo (X,Y) por otra de 3×3, mientras que es posible hacerlo si el punto está definido por tres valores (X,Y,1), como es el caso de las coordenadas homogéneas.

Sin entrar en detalles de carácter teórico, la multiplicación de (X,Y) por la matriz

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix}$$

equivale a la de (X,Y,1) por la matriz

$$\begin{vmatrix} A & B & 0 \\ C & D & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Esto permite añadir a las transformaciones vistas anteriormente la única que faltaba, es decir, la traslación. La matriz de 3×3 que hay que utilizar para este fin podemos imaginar que ha sido obtenida de la de identidad de 2×2 elementos, modificada de tal manera que se ha transformado en

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ H & K & 1 \end{vmatrix}$$

El efecto de la transformación de un punto (X,Y,1) mediante esta matriz consiste, precisamente, en la traslación del punto en cuestión, que se convierte en (X+H, Y+K, 1). En efecto, recordemos que la regla de multiplicación por una matriz de 3×3 elementos da como resultado

$$(X, Y, 1) \begin{vmatrix} A & B & 0 \\ C & D & 0 \\ H & K & 1 \end{vmatrix} = (AX+CY+H, BX+DY+K, 1)$$

En el caso de que A, B, C y D sean la matriz de identidad obtendremos precisamente el resultado (X+H, Y+K, 1), como comprobará si hace la prueba.

El elemento "H" genera, por tanto, una traslación sobre el eje "X", mientras que "K" actúa sobre el eje "Y". Es evidente que al combinar estos valores podremos efectuar cualquier traslación.

Con las coordenadas homogéneas también es posible realizar todas las transformaciones vistas anteriormente: bastará con sustituir los cuatro parámetros A, B, C y D, contenidos en la matriz de 3×3, por los de la matriz de 2×2 necesaria para efectuar esa transformación en concreto. Vamos a poner un ejemplo: una rotación hacia la izquierda de un ángulo T alrededor del origen se obtendrá con la matriz

$$\begin{vmatrix} COS(T) & SIN(T) & 0 \\ -SIN(T) & COS(T) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Experimentando con coordenadas homogéneas

Para experimentar las posibles transformaciones con coordenadas homogéneas se puede utilizar el programa 2.16.

```
10 REM *** PROGRAMA 2.16 ***
20 REM
30 REM COORDENADAS HOMOGENEAS
40 REM
50 REM
60 XI=0:XD=319:YA=0:YB=199
70 CX=160:CY=100:FE=1.1
80 REM
90 PRINT "♥":GOTO 390
100 REM
110 REM TRAZADO DE LOS EJES
120 REM
130 FOR X=0 TO XD STEP 2:PLOT X,CY,1:NEXT X
140 FOR Y=0 TO YB STEP 2:PLOT CX,Y,1:NEXT Y
150 REM
160 REM CLASIFICACION PUNTOS
170 REM
180 I(P)=X(P)<XI
190 D(P)=X(P)>XD
200 A(P)=Y(P)<YA
210 B(P)=Y(P)>YB
220 RETURN
230 REM
240 REM CALCULA LOS EXTREMOS
250 REM
```

```
260 GOSUB 180
270 V=0:IF I(1)*I(2)+D(1)*D(2)+A(1)*A(2)+
    B(1)*B(2)<>0 THEN 350
280 P=1:IF I(1)+D(1)+A(1)+B(1)<>0 THEN 310
290 P=2:IF I(2)+D(2)+A(2)+B(2)<>0 THEN 310
300 V=1:GOTO 350
310 IF I(P)<>0 THEN Y(P)=Y(1)+(XI-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XI:GOTO 260
320 IF D(P)<>0 THEN Y(P)=Y(1)+(XD-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XD:GOTO 260
330 IF A(P)<>0 THEN X(P)=X(1)+(YA-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=YA:GOTO 260
340 IF B(P)<>0 THEN X(P)=X(1)+(YB-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=XB:GOTO 260
350 RETURN
360 REM
370 REM PROGRAMA PRINCIPAL
380 REM
390 PRINT "[S]ESTE PROGRAMA MUESTRA EL EFECTO"
400 PRINT "DE UNA MATRIZ DE TRANSFORMACION"
410 PRINT "DE 3X3 ELEMENTOS"
420 PRINT "[Q]LAS COORDENADAS DE LA FIGURA ESTAN"
430 PRINT "INTRODUCIDAS EN EL PROGRAMA EN FORMA"
440 PRINT "DE LINEAS 'DATA'."
450 PRINT "[Q][Q] INTRODUCE LA MATRIZ: [Q]"
460 INPUT "A=";A:INPUT "B=";B
470 INPUT "C=";C:INPUT "D=";D
480 INPUT "H=";H:INPUT "K=";K
490 REM
500 HIRES 1,0:GOSUB 130
510 RETORE:READ NS
520 FOR SE=1 TONS:FOR P=1 TO 2:READ X1,Y1
530 XT=A*X1+C*Y1+H:YT=B*X1+D*Y1+K
540 X(P)=FE*XT+CX+.5:Y(P)=CY-YT+.5
550 GOSUB 180:NEXT P:GOSUB 270
560 IF VTHEN LINE X(1),Y(1),X(2),Y(2),1
570 NEXT SE
580 REM
590 GET T$:IF T$="" THEN 590
600 NRM
610 PRINT "[Q][Q]<ESPACIO>=BASTA,<RETURN>=OTRA VEZ";
620 GET T$
```



```
630 IF T$=CHR$(32) THEN PRINT "[Q][Q]":END
640 IF T$=CHR$(13) THEN 390
650 GOTO 620
660 REM
670 REM DESCRIPCION DE LA FIGURA
680 REM
690 DATA 21:REM No. SEGMENTOS
700 DATA 0,0,40,0,40,0,40,38,40,38,20,59
710 DATA 20,59,0,38,0,38,0,0,14,0,14,14
720 DATA 14,14,26,14,26,14,26,0,6,22,18,22
730 DATA 18,22,18,34,18,34,6,34,6,34,6,22
740 DATA 12,22,12,34,22,22,34,22,34,22,34,34
750 DATA 34,34,22,34,22,34,22,22,28,22,28,34
760 DATA 26,52,26,56,26,56,30,56,30,56,30,48
READY
```

Como verá no es otra cosa sino una versión más extensa y mejorada del anterior (2.15). Extensa porque requiere que sean introducidos también los valores H y K; mejorada porque incluye una subrutina de clipping (líneas 160-220 y 240-350) para impedir un bloqueo del programa en caso de que la imagen "se salga" de la pantalla. Además, la figura descrita en las líneas DATA que pueden ser modificada como se quiera, es más compleja que el cuadrado utilizado anteriormente; se trata en este caso de una casita con una puerta, ventanas y chimenea (Fig. 2). Como ejemplo de utilización del programa, la figura 3 muestra la traslación de la imagen, hecha posible gracias a la adopción de coordenadas homogéneas.

Transformaciones más complejas

Nuestro propósito es el de obtener con una sola operación el equivalente de una serie de transformaciones efectuadas una tras otra. Como en el caso anterior, para hacer esto tendremos que multiplicar entre sí las matrices correspondientes a las transformaciones independientes, teniendo cuidado de efectuar los productos en el orden adecuado (recuerde que aquí no es válida la propiedad conmutativa).

La regla para multiplicar dos matrices de 3×3 elementos es la siguiente:

$$\begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix} \begin{vmatrix} J & K & L \\ M & N & O \\ P & Q & R \end{vmatrix} = \begin{vmatrix} AJ+BM+CP & AK+BN+CQ & AL+BO+CR \\ DJ+EM+FP & DK+EN+FQ & DL+EO+FR \\ GJ+HM+IP & GK+HN+IQ & GL+HO+IR \end{vmatrix}$$

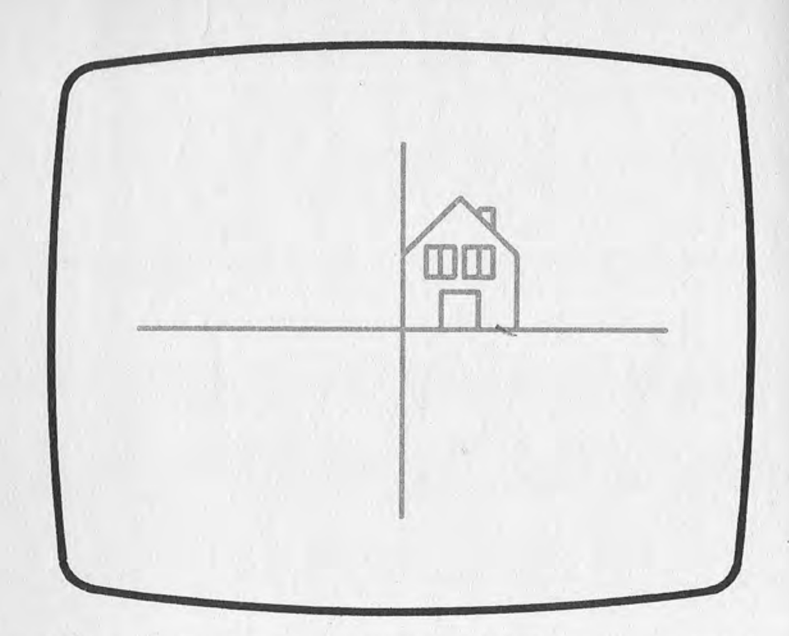


Figura 2.—La casita definida en los DATA del programa 2.16. Al utilizar una matriz adecuada la imagen puede ser transformada en una gran variedad de maneras.

Resulta un poco enrevesada a la hora de aplicarla, pero procediendo con calma podrá evitar cometer errores.

Pongamos, por ejemplo, que queremos rotar la casita del programa 2.16 hacia la izquierda 30 grados y que, aprovechando las posibilidades que las coordenadas homogéneas nos ofrecen, queremos que tal rotación no se produzca alrededor del origen, sino que se halle centrada en un punto arbitrario que escogemos en este caso, como el de coordenadas (-10,20). Esto equivale a efectuar la secuencia de transformaciones siguientes:

— hacer una traslación tal que el punto (-10,20) que será el centro de rotación, se convierta en el origen de los ejes cartesianos. Obtenemos esto por medio de una traslación del eje "X" de 10 puntos y otra del eje "Y" de -20 puntos;
— efectuar una rotación, centrada en el nuevo origen, de 30 grados en sentido opuesto al de las manecillas de un reloj;
— volver a llevar a su sitio el centro de rotación, con una traslación de -10 sobre el eje "X" y de 20 sobre el eje "Y".

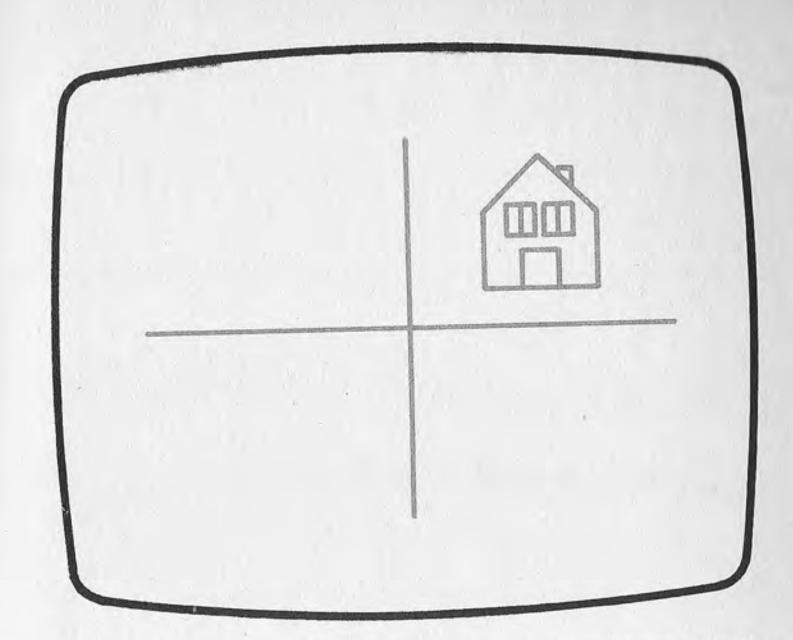


Figura 3.—A diferencia de lo que ocurre con las coordenadas cartesianas o polares, con las homogéneas es posible efectuar una traslación de la imagen, como aparece en esta figura.

La matriz que realiza las transformaciones primera y tercera es la de traslación, mientras que para la segunda es necesaria una matriz de rotación. Por tanto se tratará de multiplicar entre sí estas matrices:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ H & K & 1 \end{vmatrix} \quad \begin{vmatrix} COS(T) & SIN(T) & 0 \\ -SIN(T) & COS(T) & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ H & K & 1 \end{vmatrix}$$

Sustituyendo los valores del ejemplo:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & -20 & 1 \end{vmatrix} \quad \begin{vmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -10 & 20 & 1 \end{vmatrix}$$

Tras haber efectuado las multiplicaciones con calma (la primera matriz por la segunda y luego el resultado por la tercera) se obtiene una única matriz de 3×3:

$$\begin{vmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 8.6 & 7.68 & 1 \end{vmatrix}$$

que, aplicada a la figura, dará origen a la transformación deseada (Fig. 4). El programa 2.16 permite comprobar el efecto de esta matriz sobre la imagen de la casita.

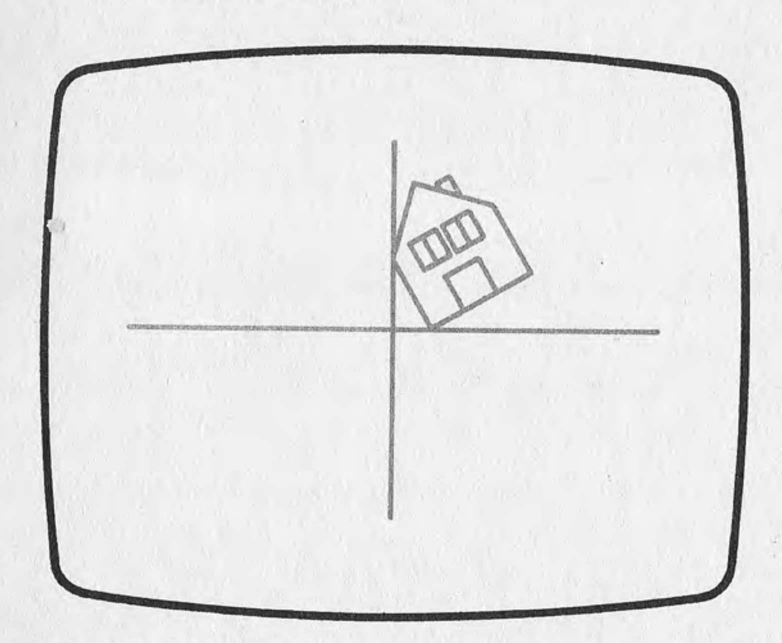


Figura 4.—Al calcular adecuadamente la matriz de transformación la casita podrá sufrir una rotación alrededor de un punto cualquiera, que no tiene por qué coincidir con el centro de los ejes. En este caso la rotación es de 30 grados en sentido antihorario, alrededor del punto de coordenadas (-10,20).

Es interesante observar que al multiplicar entre sí matrices del tipo de las utilizadas para trabajar con coordenadas homogéneas se obtienen como resultado matrices del mismo tipo. En otras palabras, el resultado será siempre de la forma:

$$\begin{vmatrix} A & B & 0 \\ C & D & 0 \\ H & K & 1 \end{vmatrix}$$

en la que la columna de la derecha contiene los valores constantes 0,0,1.

Representar una función

Vamos a dejar por un momento de lado las transformaciones geométricas para dedicarnos a la representación gráfica de funciones. Entendemos aquí por función el caso más sencillo: una relación matemática que liga una variable, llamada dependiente, al valor asumido por otra, llamada por esto variable independiente. Indicaremos la primera con la letra "Y" y la segunda con "X". Cuando "X" varía lo hará también "Y", dependiendo precisamente de la función elegida.

Un ejemplo bastante familiar es el de la función seno de un ángulo, en dependencia del valor de éste, mostrado en la figura 5. La figura se obtiene al representar sobre el eje "X" el valor del ángulo, y sobre el eje "Y" el valor del seno (es decir, de la función); se dice entonces que el seno está en función del ángulo.

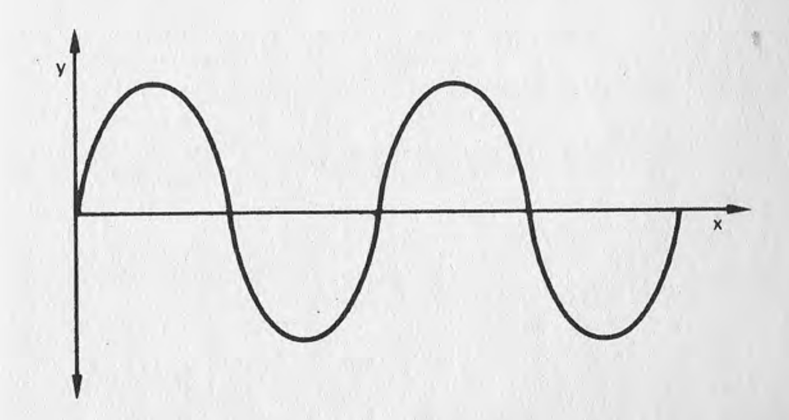


Figura 5.—Ejemplo de función: en este caso se trata de una sinusoide de ecuación Y=SIN(X).

```
10 REM *** PROGRAMA 2.17 ***
20 REM
30 REM REPRESENTAR UNA FUNCION
40 REM
50 REM
60 CY=100:MX=319
70 REM
80 REM FUNCION QUE HAY QUE CALCULAR
90 REM
100 DEF FNA(X)=1/EXP(X*X)
110 GOTO 230
120 REM
130 REM TRAZADO EJE X
140 REM
150 FOR X=0 TO MX STEP 2:PLOT X,CY,1:NEXT X
160 FOR X=0 TO MX STEP 10
170 FOR Y=CY-4 TO CY+4 STEP 2
180 PLOT X,Y,1:NEXT Y,X
190 RETURN
200 REM
210 REM PROGRAMA PRINCIPAL
220 REM
230 PRINT "▼ESTE PROGRAMA TRAZA EL GRAFICO"
240 PRINT "DE UNA FUNCION Y=F(X)."
250 PRINT "Q LA FUNCION DEBE ESTAR DEFINIDA EN LA"
260 PRINT "LINEA 100;ACTUALMENTE ES:"
270 PRINT "QQQQQQQ TECLEA EL COMANDO 'GOTO 330'
    SI ES CORRECTA,"
280 PRINT "EN CASO CONTRARIO SUSTITUYELA Y LUEGO
    TECLEA"
290 PRINT "DE NUEVO 'RUN' ●●●●●●●●● "
300 REM
310 LIST 100:REM IMPRIME LA FUNCION ACTUAL
320 REM
330 INPUT "▼QQQQ VALOR MINIMO DE X";X1
340 INPUT " VALOR MAXIMO DE X";X2
350 INPUT "Q ¿CUANTOS PUNTOS HAY QUE CALCULAR?";NP
360 REM
370 PRINT "▼QQQQQQ CALCULO DE LA FUNCION"
380 PRINT "Q      PARA"
390 PRINT "Q BUSQUEDA DE MINIMO Y MAXIMO"
```



```
400 REM
410 M=0:S=MX/NP:DX=(X2-X1)/MX
420 FOR X=0 TO MX+.5 STEP S:Y=FNA(X1+X*DX)
430 IF ABS(Y)>M THEN M=ABS(Y)
440 NEXT X
450 NO=(CY-2)/M
460 REM
470 REM TRAZADO FUNCION
480 REM
490 HIRES 1,0:GOSUB 150
500 FOR X=0 TO MX+.5 STEP S
510 Y=CY-FNA(X1+X*DX)*NO+.5
520 IF X=0 THEN 540
530 LINE XP,YP,X,Y,1
540 XP=X:YP=Y:NEXT X
550 REM
560 GET T$:IF T$="▼" THEN 560
570 PRINT " "
READY
```

El programa 2.17, con el que finaliza la sección dedicada a gráficos en dos dimensiones, tiene como fin la representación gráfica de una función cualquiera elegida por el usuario. No son necesarias explicaciones especiales, pues las técnicas empleadas han sido utilizadas en su mayoría en otros programas anteriores. Tan sólo hay ciertos puntos que merecen ser señalados:

- la función que hay que trazar está definida en la línea 100 en forma de FNA(X), y por tanto es fácilmente modificable. La que aparece en el listado corresponde a una curva llamada gaussiana o "de campana";
- para mayor claridad ha sido dibujado sólo el eje X, como una línea discontinua de referencia;
- el programa solicita los valores mínimos y máximo entre los que podrá variar "X" para dibujar la función. Por tanto, será posible elegir cada vez el intervalo que se desee: para la curva del ejemplo puede hacerse la prueba con "X" comprendido entre -4 y 4;
- hay que definir también el número de puntos en los que será subdividido el intervalo donde se mueve la variable "X"; naturalmente, cuantos más puntos hayan sido utilizados en el cálculo más preciso será el dibujo obtenido. Para la curva introducida en el listado pueden ser suficientes 50 puntos;

— el programa efectúa un primer cálculo de la función en todo el intervalo indicado, pero sin dibujar nada (líneas 420-440). El objetivo es el de hallar el valor máximo (tanto positivo como negativo) alcanzado por la función (línea 430);
— referido al valor máximo se obtiene un "factor de normalización" NO (línea 450). La función es calculada de nuevo (líneas 500-540) y el factor NO interviene en el cálculo para asegurar que aquélla no se sale en ningún punto del espacio útil de la página gráfica;
— en las líneas 420 y 500 el límite del bucle de cálculo no es MX, sino MX+0.5.

Esta corrección es útil porque de otro modo, debido al pequeño error de aproximación que se produce inevitablemente en cada suma (es decir en el NEXT X), la variable "X" alcanzaría el valor final con una anticipación de un ciclo respecto al número de iteraciones previstas (como ocurría en el programa 2.14).

Obsérvese, por último, que el programa 2.17, al no utilizar ninguna graduación de los ejes, sirve tan sólo para dar una idea cualitativa de la función que se está estudiando. Además, no se establece ninguna protección contra un error típico que puede ocurrir durante el cálculo: el que en un punto el valor de la función tienda al infinito, como puede ocurrir si, al calcular el valor de la coordenada "X" de ese punto, el programa trata de efectuar una división por cero.

He aquí una breve lista de funciones que pueden ser experimentadas:

— FNA(X)=SIN(X), con X comprendido entre 0 y 6.28 (¡en radianes!);
— FNA(X)=SIN(X)/X, con X comprendido entre 0 y 25.13 (es decir 8PI);
— FNA(X)=X*X (o, lo que es lo mismo, X↑2), con X comprendido entre -10 y 10.

Cada una de estas funciones puede ser trazada de una forma aceptable con 50 puntos.

CAPITULO VI

GRÁFICOS EN TRES DIMENSIONES

De todas las técnicas tratadas en este libro, las relativas a gráficos tridimensionales representan, sin duda alguna, el aspecto más intresante. Se trata de superar las limitaciones impuestas por la pantalla, de dos dimensiones, y representar en perspectiva imágenes de sólidos o funciones en tres dimensiones. Para obtener este resultado no se necesita nada particular: basta proceder con orden y seguir atentamente la teoría expuesta a continuación.

Las técnicas que utilizaremos requieren el conocimiento de algunos conceptos geométricos y matemáticos que no podremos demostrar muy ampliamente en este libro. De todas formas, las explicaciones que van incluidas son suficientes para entender los programas, e incluso modificarlos, a fin de hacer experimentos, cosa que aconsejamos decididamente.

Coordenadas de un punto en el espacio

Para localizar un punto sobre un plano hemos utilizado dos métodos diferentes: las coordenadas cartesianas y las polares. Al pasar ahora al espacio será necesario añadir un tercer valor a las coordenadas cartesianas, ya que nos hallamos con una dimensión más (Fig. 1); así que ahora cada punto se identificará con tres números: X,Y,Z.

Lo mismo vale en relación con las coordenadas polares, que toman en este caso el nombre de coordenadas esféricas. La figura

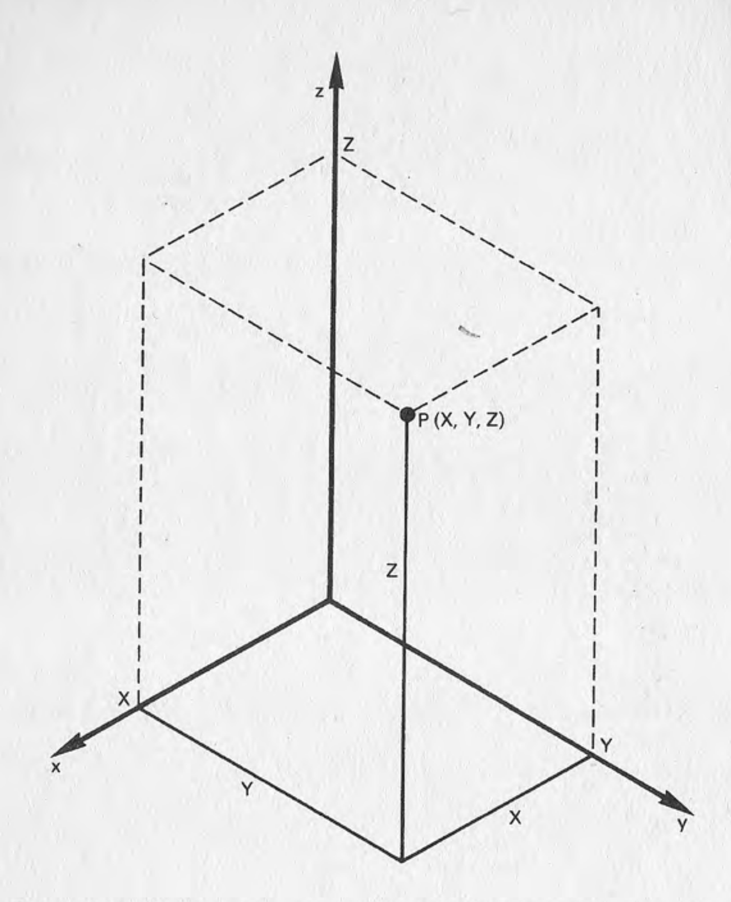


Figura 1.—El espacio ya no está referido a dos, sino a tres ejes cartesianos. Para localizar la posición de un punto P se necesitarán, por tanto, tres coordenadas: X,Y,Z.

2 muestra que también con estas coordenadas es necesario utilizar tres números (dos ángulos y una distancia) para identificar un punto.

Representación en perspectiva

Nuestro propósito es obtener en la pantalla una imagen de un sólido tridimensional visto en perspectiva. Como es lógico ese resultado se puede obtener utilizando una serie de técnicas dife-

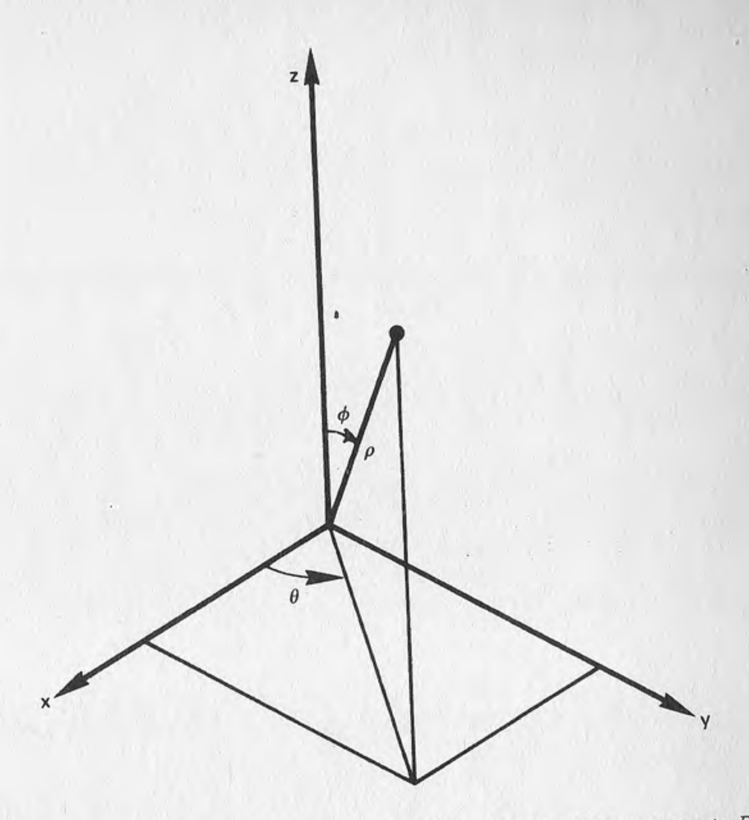


Figura 2.—Si se utilizan coordenadas esféricas el mismo punto P de la figura 1 estará definido por los ángulos θ (theta) y ϕ (phi) y por la distancia ρ (rho) al origen. El ángulo θ se mide, en sentido contrario al de las agujas de un reloj, en el plano xy a partir del eje x, mientras que ϕ se mide entre el eje z y la recta que une el origen con P. ρ es la distancia de P al origen.

rentes entre sí. Nosotros emplearemos una de ellas en concreto, que consiste en seguir uno tras otro los siguientes pasos:

1. Se establece en primer lugar la posición del observador en relación con los ejes cartesianos. En este caso resulta más cómodo utilizar las coordenadas esféricas, así que esa posición (que llamaremos "P") se define mediante los valores de los ángulos θ (theta) y ϕ (phi) y de la distancia al origen ρ (rho) (Fig. 2). Es evidente que al observar un objeto tridimensional bajo puntos de vista diferentes, su representación en perspectiva también variará.

2. Se transforman las coordenadas X,Y,Z de cada punto de forma que están referidas, no ya a los ejes cartesianos, que llamamos habitualmente x,y,z, sino a una nueva terna de ejes cuyo origen coincide con la posición del observador. Indicaremos estos nuevos ejes como x',y',z'. La figura 3 puede servir de ayuda para entender esta situación.
3. Por medio de operaciones de rotación se orientan los ejes x',y',z' de forma que z' se halle dirigido hacia el origen del sistema cartesiano de partida y que el eje x' esté en posición horizontal y dirigido hacia la derecha (mirando hacia el origen), mientras que y' debe estar vertical y hacia arribal (para lograr esta combinación debemos cambiar de signo el eje x" resultante de las rotaciones). Asignaremos a la nueva terna de ejes el nombre de xo, yo, zo (Fig. 4).
4. Por último se proyecta cada punto del objeto que hay que representar sobre un plano paralelo a (xo, yo). Al señalar en la página gráfica la posición de los puntos proyectados y unirlos luego entre sí en el orden correcto (como ya ocurría por las técnicas anteriores) se obtiene finalmente la representación en perspectiva del objeto en cuestión (Figura 5).

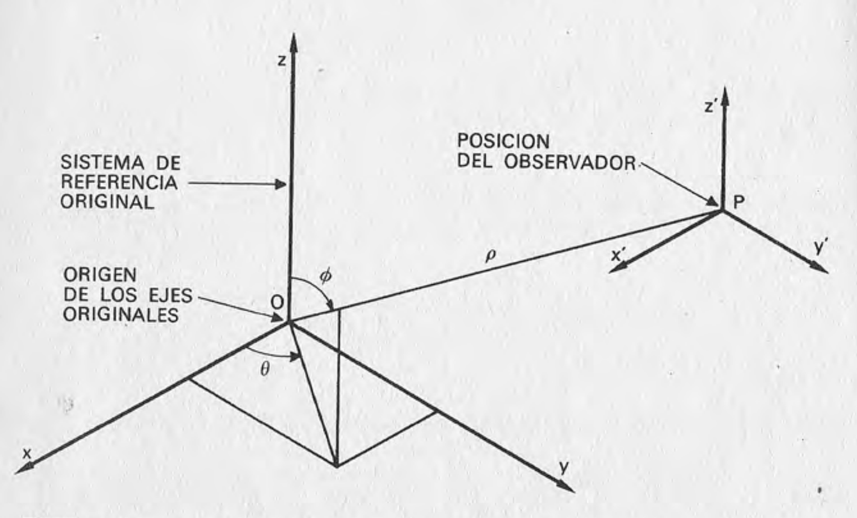


Figura 3.—La posición del observador, "P", se define en el habitual sistema de ejes mediante sus coordenadas esféricas θ,φ,ρ. En este punto se sitúa el origen de un nuevo sistema de referencia, cuyos ejes x',y',z' son paralelos a los originarios x,y,z.

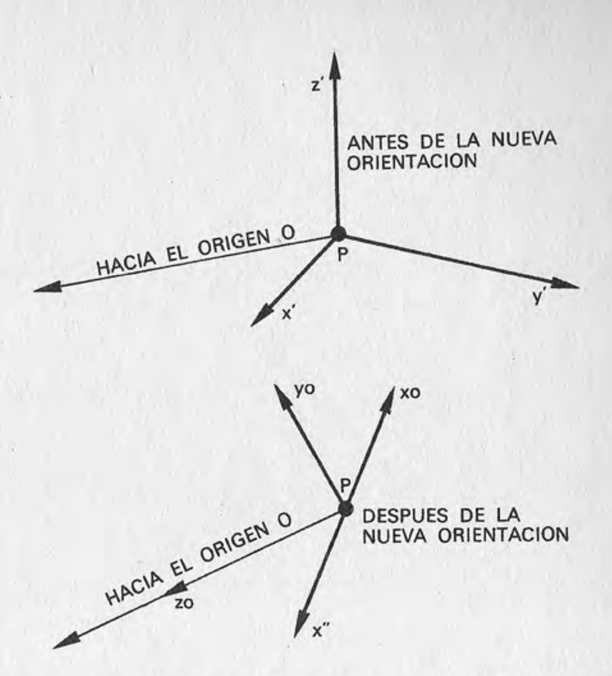


Figura 4.—Al aplicar dos rotaciones sucesivas el sistema de referencia x',y',z' (paralelo a x,y,z) se transforma en otro sistema diferente xo,yo,zo.

Vamos a examinar ahora uno por uno los cuatro pasos de la técnica de representación tridimensional utilizando un programa.

```
10 REM *** PROGRAMA 3.1 ***
20 REM
30 REM OBJETO TRIDIMENSIONAL
40 REM
50 REM
60 CX=160:CY=100:FE=1.1
70 RH=5:TH=.5:PH=1:D=400
80 S1=SIN(TH):C1=COS(TH)
90 S2=SIN(PH):C2=COS(PH)
100 HIRES 1,0:GOTO 250
110 REM
120 REM CALCULO COORDENADAS VISUALIZACION
130 REM
140 XO=-X*S1+Y*C1
```

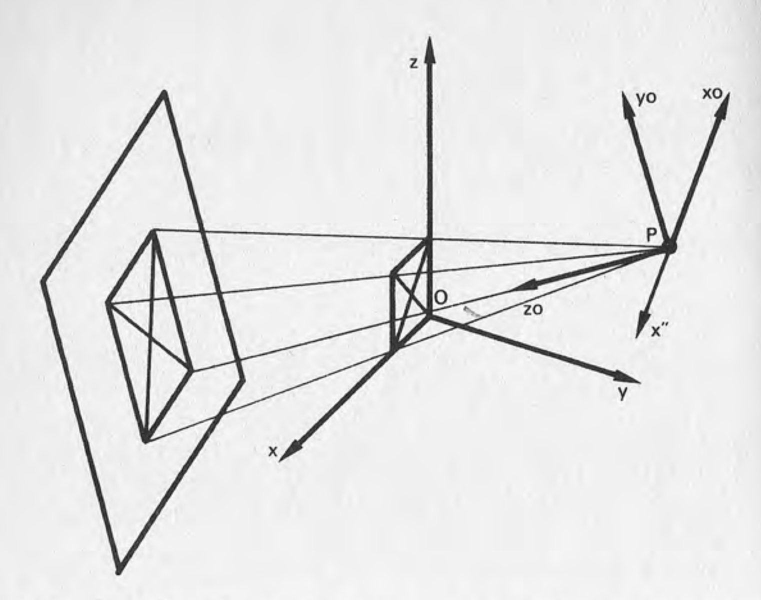


Figura 5.—Para obtener la imagen en perspectiva se proyecta cada punto de la figura originaria, en este caso un cuadrado con sus diagonales, sobre un plano paralelo al formado por los ejes "xo" e "yo". Para que resulte más sencillo, en el ejemplo ha sido proyectada una figura bidimensional (el cuadrado), pero el mismo concepto valdrá también para sólidos.

```
150 YO=-X*C1*C2-Y*S1*C2+Z*S2
160 ZO=-X*S2*C1-Y*S2*S1-Z*C2+RH
170 XV=.5+CX+D*XO/ZO*FE
180 YV=.5+CY-D*YO/ZO
190 RETURN
200 REM
210 REM PROGRAMA PRINCIPAL
220 REM
230 REM EJES X,Y,Z
240 REM
250 FOR I=1 TO 3
260 READ X,Y,Z:GOSUB 140:XP=XV:YP=YV
270 READ X,Y,Z:GOSUB 140:LINE XP,YP,XV,YV,1
280 NEXT I
290 REM
300 REM CUBO
```



```
310 REM
320 FOR I=1 TO 2:FOR J=1 TO 5
330 READ X,Y,Z:GOSUB 140
340 IF J=1 THEN 360
350 LINEA XP,YP,XV,YV,1
360 XP=XV:YP=YV:NEXT J,I
370 FOR I=1 TO 4
380 READ X,Y,Z:GOSUB 140:XP=XV:YP=YV
390 READ X,Y,Z:GOSUB 140:LINE XP,YP,XV,YV,1
400 NEXT I
410 REM
420 GET T$:IF T$="" THEN 420
430 END
440 REM
450 REM DEFINICION DE LA FIGURA
460 REM
470 DATA 1,0,0,1.5,0,0
480 DATA 0,1,0,0,1.5,0
490 DATA 0,0,1,0,0,1.2
500 DATA 1,0,0,1,0,1,1,1,1,1,1,0,1,0,0
510 DATA 0,0,0,0,0,1,0,1,1,0,1,0,0,0,0
520 DATA 1,0,0,0,0,0,1,0,1,0,0,1
530 DATA 1,1,0,0,1,0,1,1,1,0,1,1
READY
```

Primer paso: fijar la posición del observador

El punto desde donde miramos un objeto tridimensional del que queremos obtener una imagen en perspectiva puede definirse como se quiera. Basta con asignar a tres variables las tres coordenadas esféricas de la posición del observador. En el programa 3.1 son utilizados RH, TH y PH (línea 70), que representan, respectivamente, la distancia al origen "ρ", el ángulo horizontal "θ" y el vertical "ф" del punto "P".

El motivo por el cual se utilizan coordenadas esféricas, en lugar de las habituales rectangulares, es el de hacer más sencillo tanto el desplazamiento arbitrario del punto "P" como los cálculos que seguirán.

Refiriéndonos de nuevo a la figura 3, donde la posición del observador es indicada como "P", notamos que un ángulo θ igual a cero significa que aquél ve el objeto a lo largo de la dirección del eje "X". Si, en cambio, el ángulo ф vale cero, el punto de vista se

halla por encima del objeto, en la dirección del eje "Z". En la línea 70 se asigna también un valor a la variable "D", de la que hablaremos más adelante.

Segundo paso: transformación de las coordenadas en el nuevo sistema de referencia

Sea un punto genérico situado en el espacio, que indicaremos como "A". Utilizando un sistema de referencia cartesiano, cuyos ejes sean x,y,z la posición del punto está definida por la terna de coordenadas X,Y,Z, así que llamaremos este punto A(X,Y,Z).

Imaginemos ahora que hacemos una operación un tanto extraña: dejamos inalterada la posición del punto "Y" en el espacio, pero desplazamos los ejes cartesianos. Este desplazamiento es tal que no se produce rotación alguna, y los nuevos ejes se mantienen paralelos a los anteriores. Obtendremos así otro sistema de referencia cuyos ejes recibirán el nombre de x',y',z' (Fig. 6).

Pero volvamos a nuestro punto "A", cuya posición en el sistema de referencia originario está dada por los tres valores X,Y,Z. La posición de este mismo punto "A" se podrá referir también a la nueva terna de ejes x',y',z' sin que, obviamente, se produzca variación alguna del punto. Obtendremos entonces tres coordenadas, naturalmente diferentes que las primeras, que esta vez indicaremos con X',Y',Z'.

Pero atención: ya hemos dicho que el punto "A" no se ha movido durante la operación de traslación de los ejes, luego como la medida de ese desplazamiento es conocida, debe ser posible calcular las nuevas coordenadas X',Y',Z' a partir de las antiguas (X,Y,Z).

Para medir el desplazamiento sufrido por los ejes indicaremos con "P" la posición del nuevo origen, medida siempre respecto del sistema x,y,z (Fig. 6). Esa posición se identifica con tres coordenadas, que llamaremos XP, YP, ZP. No resulta entonces difícil deducir que las coordenadas del punto "A" en el nuevo sistema de referencia son:

X'=X-XP   Y'=Y-YP   Z'=Z-ZP

Pero... un momento: esto es sólo una sencilla operación de traslación de un punto a lo largo de un eje, parecido al que ya vimos al tratar de figuras bidimensionales; en realidad, el desplazamiento del origen de un sistema de referencia produce el mismo resultado que el desplazamiento de la posición del punto "A" una distancia opuesta.

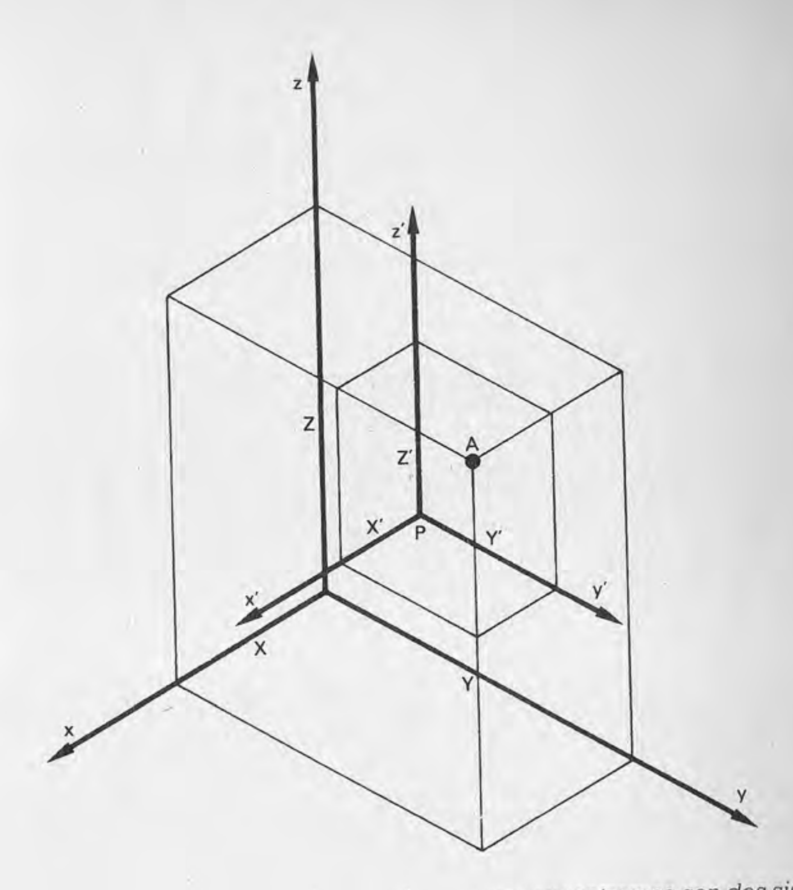


Figura 6.—Al realizar el segundo paso nos encontramos con dos sistemas de referencia cuyos ejes son paralelos. En esta situación todo punto "A" de coordenadas conocidas X,Y,Z podrá ser fácilmente referido al nuevo sistema x',y',z' por medio de las coordenadas X',Y',Z'.

Al tratarse de una simple traslación, para introducirla en nuestros cálculos utilizaremos una vez más una matriz de transformación. Vamos a explicar de qué manera: como ya dijimos "P" es la posición del observador, que hemos expresado en coordenadas esféricas. Como para efectuar la traslación necesitamos coordenadas rectangulares tendremos que hacer una transformación. Vamos a definir en primer lugar cuatro nuevas variables (líneas 80 y 90) haciendo uso de las utilizadas anteriormente:

S1=SIN(TH)   C1=COS(TH)
S2=SIN(PH)   C2=COS(PH)

Utilizaremos nuevamente una regla de trigonometría que establece que las coordenadas rectangulares de cualquier punto, por ejemplo el "P", pueden obtenerse simplemente a partir de las esféricas mediante las relaciones siguientes (que no vamos a demostrar en este libro):

XP=RH*C1*S2
YP=RH*S1*S2
ZP=RH*C2

Recuerde que la terna XP, YP, ZP señala la posición del observador en el sistema de ejes cartesianos inicial x,y,z.

Construimos por último una matriz de traslación, que será utilizada para desplazar la posición de cada punto una cantidad OPUESTA a la que habíamos dicho. En el espacio tridimensional las coordenadas homogéneas poseen, naturalmente, cuatro términos, así que la matriz de traslación estará en este caso constituida por:

1	0	0	0
0	1	0	0
0	0	1	0
H	K	L	1

que al ser multiplicada por un punto genérico (X,Y,Z,1) lo desplaza a una nueva posición dada por

(X+H, Y+K, Z+L, 1)

Tras combinar los resultados obtenidos hasta ahora, la transformación que se requiere para cumplir el segundo paso se realiza aplicando la matriz

1	0	0	0
0	1	0	0
0	0	1	0
-RH*C1*S2	-RH*S1*S2	-RH*C2	1

con la cual desplazamos cualquier punto en -XP, -YP, -ZP.

Tercer paso: reorientación de los ejes

La tercera fase de esta técnica consiste en una nueva modificación del sistema de referencia: no se trata esta vez de una traslación, sino de una rotación de los ejes. La figura 4 muestra claramente cómo se presenta el sistema cartesiano antes y después de la nueva transformación. Por qué tenemos que recurrir a esta rotación se explica rápidamente: queremos que el siguiente paso, es decir la proyección de la imagen sobre un plano, resulte más sencilla.

Empezamos con una rotación del sistema de referencia alrededor del eje z' en el sentido de las agujas de un reloj (mirándolo desde arriba). El ángulo de rotación debe ser tal que lleve el eje x' a una posición perpendicular al segmento que une P con el origen O. Observando la figura 7 comprenderemos que se necesita una rotación de 90-θ grados.

En el párrafo anterior, al referirnos a las traslaciones, habíamos comentado un principio básico: una transformación del sistema de referencia modifica las coordenadas de un punto cualquiera del mismo, de modo que éstas son modificadas como si aplicáramos al punto una transformación igual, pero de sentido opues-

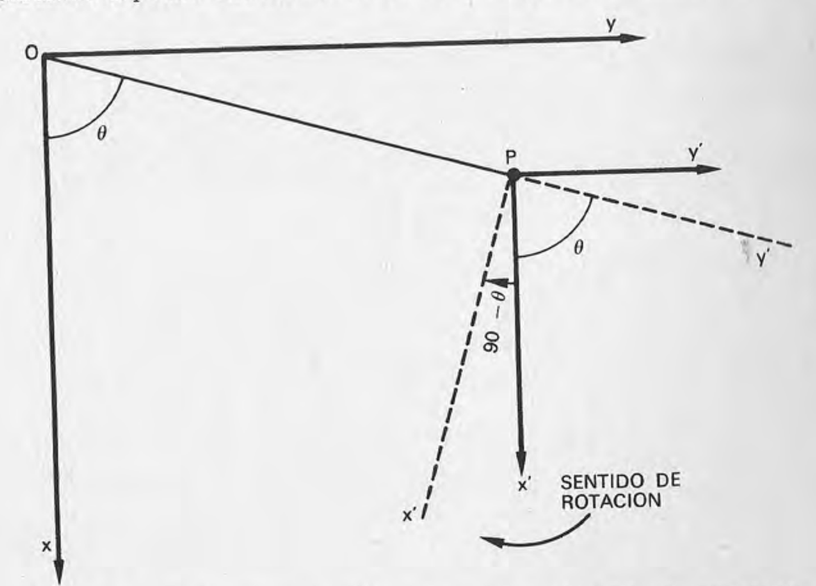


Figura 7.—La situación tal y como se ve vista "desde arriba" (por esos los ejes z,z' son invisibles). El ángulo de rotación necesario para llevar los ejes x',y' a la posición indicada por la línea discontinua es de 90-θ grados.

to. Esta regla es general y resulta por tanto válida tanto en traslaciones como en rotaciones.

Es posible hacer que un punto rote un ángulo "A" alrededor del eje "z" (en sentido contrario a las agujas del reloj) utilizando una matriz de transformación. La teoría, de la que no daremos aquí demostración, indica que la matriz que hay que aplicar es:

$$\begin{vmatrix} COS(A) & SIN\ (A) & 0 & 0 \\ -SIN(A) & COS(A) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Aplicando la regla anterior esta matriz permite también que todo el sistema de referencia gire en el sentido de las agujas del reloj un mismo ángulo (transformación opuesta), que es precisamente lo que necesitamos.

Decíamos que la rotación necesaria era de 90-θ grados. Disponemos de dos relaciones trigonométricas que determinan:

SIN(90-A)=COS(A) COS(90-A)=SIN(A)

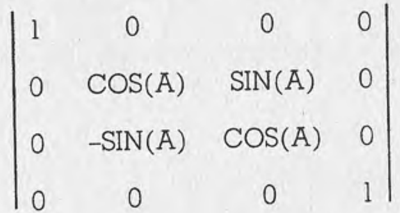
Veamos cómo afecta esto a la matriz anterior. El primer elemento es COS(A); en nuestro caso el ángulo de rotación "A" vale 90-θ grados; luego este elemento se convierte en COS(90-TH) que, por una de las relaciones precedentes, sabemos que es igual a SIN(TH). Al hacer lo mismo en toda la matriz ésta se convierte en:

$$\begin{vmatrix} SIN(TH) & COS(TH) & 0 & 0 \\ -COS(TH) & SIN(TH) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

O también, usando las variables BASIC definidas para este programa:

$$\begin{vmatrix} S1 & C1 & 0 & 0 \\ -C1 & S1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

La segunda rotación de que hemos hablado está ilustrada en la figura 8. Se trata de rotar el sistema de ejes un ángulo de 180-ϕ grados alrededor del eje "X'" (en sentido opuesto al de las agujas del reloj). La matriz que efectúa la rotación de un punto un ángulo A alrededor del eje "X" en sentido antihorario es:



$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & COS(A) & SIN(A) & 0 \\ 0 & -SIN(A) & COS(A) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

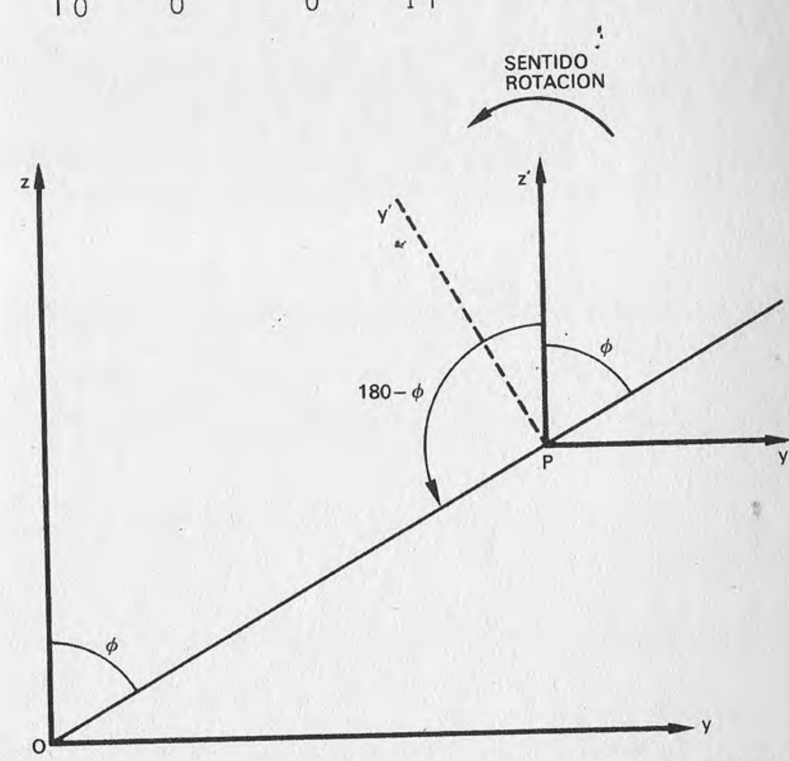


Figura 8.—La situación está vista "de lado", de forma que el eje "x'" sea invisible ("x'" no es ya perpendicular al plano del papel, pero no ha sido dibujado para mayor claridad). Lo que queremos es hacer rotar el sistema x',y',z' alrededor del eje x' hasta que z' coincida con el segmento que une O con P. Para hacer esto se necesita una rotación, contraria a la de las agujas de un reloj, de 180-ϕ grados. Después de la rotación, el eje "y'" estará en la posición indicada por la línea discontinua.

Esta misma matriz servirá para girar los ejes en el sentido opuesto (horario), pero como lo que deseamos es girarlos en el antihorario deberemos usar la matriz:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & COS(A) & -SIN(A) & 0 \\ 0 & SIN(A) & COS(A) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

que gira los ejes en sentido antihorario.

También en este caso utilizamos unas relaciones trigonométricas que permitirán simplificar el trabajo:

SIN(180-A)=SIN(A) COS(180-A)=-COS(A)

dado que, tal y como indica la figura 8, el ángulo que se necesita es de 180-θ grados, la matriz para esta segunda rotación se convierte en:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -COS(PH) & -SIN(PH) & 0 \\ 0 & SIN(PH) & -COS(PH) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -C2 & -S2 & 0 \\ 0 & S2 & -C2 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

donde la matriz de la derecha utiliza las variables del programa 3.1.

Si aplicamos a cada punto las matrices vistas hasta ahora, el sistema de referencia se transformará en el que aparece en las figuras 4 y 5 como x", yo, zo, donde el eje x" estará dirigido en sentido opuesto al que queremos, es decir, al de xo. Lo que tenemos que hacer ahora es situar correctamente el eje x". Esta operación consiste tan sólo en cambiar el signo en la coordenada X de todos los puntos y esto puede hacerse aplicando una matriz que se parece mucho a la de identidad:

$$\begin{vmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

De esta manera hemos obtenido por fin el sistema de referencia deseado, según la posición del observador; los ejes se indican con xo,yo,zo.

Cálculo de la transformación resultante

Ya sabemos que es posible efectuar una transformación complicada sirviéndonos de una sola matriz, con tal de que ésta sea el producto de distintas matrices que corresponden a las transformaciones elementales que queremos aplicar. En este caso las transformaciones que hay que efectuar en secuencia son cuatro: una traslación, dos rotaciones y una simetría.

A pesar de no ser difícil, el hecho de multiplicar cuatro matrices de 4×4 elementos cada una resulta bastante complejo, así que no vamos a desarrollarlo aquí. Nos bastará, para nuestro objetivo, conocer la matriz resultante que, como puede verse, no es particularmente compleja:

$$\begin{vmatrix} -S1 & -C1*C2 & -C1*S2 & 0 \\ C1 & -S1*C2 & -S1*S2 & 0 \\ 0 & S2 & -C2 & 0 \\ 0 & 0 & RH & 1 \end{vmatrix}$$

Además de los senos y cosenos de los ángulos θ y ϕ, puede verse que en la matriz aparece también la distancia del observador al origen, que hemos indicado como ρ (variable RH). La transformación de cada punto se obtiene, como ya sabemos por los capítulos anteriores, al multiplicar las coordenadas del punto por la matriz de transformación (las reglas para hacerlo se explican más adelante). En el programa es efectuada por las líneas 140-160; las coordenadas resultantes son asignadas a las variables XO,YO,ZO (recuérdese que la letra "O" significa "observador").

Cuarto paso: proyección sobre un plano de cada punto de la figura

Lo que queremos obtener con la cuarta y última fase de esta técnica aparece en la figura 5: proyectar cada punto del objeto sobre un plano para obtener una imagen en perspectiva. El plano debe ser perpendicular al segmento que une el observador "P" y

el origen "O", paralelo por tanto al plano XO,YO; se trata de una superficie situada enfrente de quien está mirando. Será la pantalla del monitor de vídeo.

Esta operación significa que tendremos que asignar a cada punto, de coordenadas X,Y,Z, del sólido originario otro correspondiente en el plano de proyección. Cada uno de estos puntos estará definido por sólo dos coordenadas, que llamaremos XV e YV. En este caso la letra "V" indica que se trata de coordenadas para la visualización.

El mecanismo de la proyección aparece en la figura 9. El sistema de referencia es el relativo al observador (xo,yo,zo) ya que, tras haber aplicado la matriz de transformación (obtenida por los tres primeros pasos del procedimiento) cada punto XO,YO,ZO estará referido con relación a este sistema. La figura está vista "desde arriba", de ahí que el eje "yo" no sea visible y el plano de proyección aparezca cortado.

Sea un punto genérico "A" del que queremos obtener la proyección correspondiente "A'" en el plano de proyección. En el sistema de ejes utilizado, las coordenadas de "A" serán XO,YO,ZO (en la figura aparecen sólo XO y ZO). Nótese que los triángulos $\widehat{PBA}$ y $\widehat{PB'A'}$ (ambos rectángulos) son semejantes, así que será cierta la siguiente relación matemática:

$$\frac{BA}{BA'} = \frac{PB}{PB'} = \frac{PA}{PA'}$$

Recordando que la distancia del plano de proyección al observador ha sido indicada con la variable "D" y sustituyendo en la relación anterior las coordenadas del punto "A" y de su proyección "A'", obtenemos:

$$\frac{XO}{XV} = \frac{ZO}{D}$$

de donde se deduce que la coordenada "X" de la proyección "A'" del punto "A" vale

XV=D*XO/ZO

Para obtener la otra coordenada, es decir YV, se hará lo mismo, pero observando esta vez la situación "de lado" (en este caso el que no será visible será el eje "xo"). Las relaciones geométricas y matemáticas son las mismas que las que hemos visto ahora y dan como resultado:

YV=D*YO/ZO

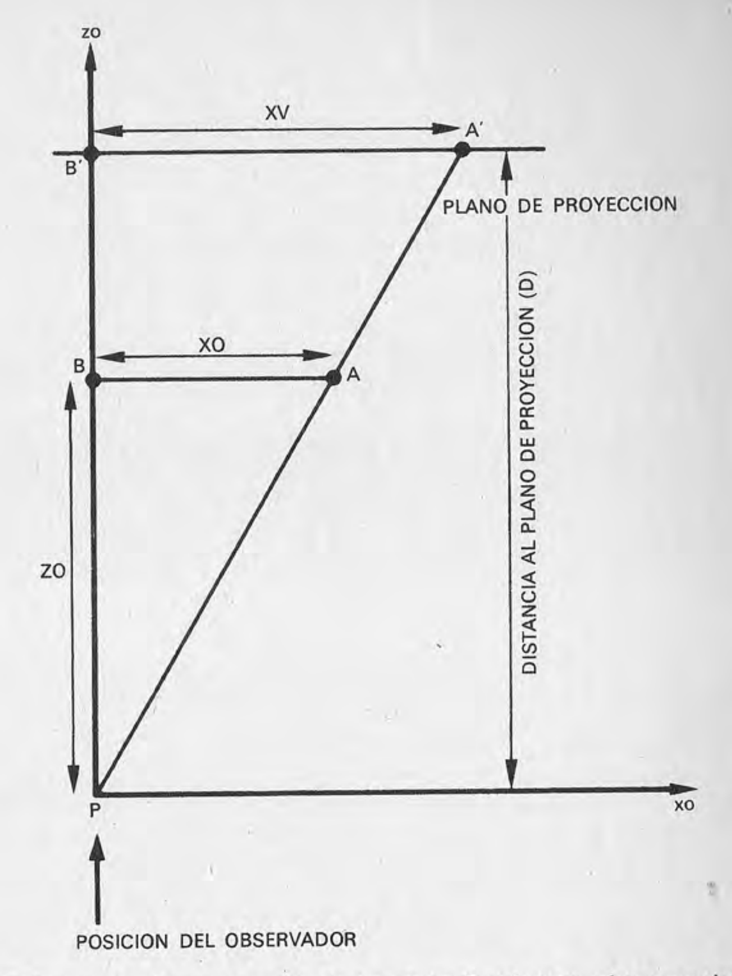


Figura 9.—La proyección de un punto A sobre un plano paralelo al formado por los ejes "x" e "y" y situado a una distancia D de éste, se halla aplicando las reglas de los triángulos semejantes. En la figura se muestra el procedimiento válido para las coordenadas X (el eje "yo" "sale" perpendicularmente del papel). Los triángulos $\widehat{BPA}$ y $\widehat{B'PA'}$ son semejantes.

Estos valores XV e YV no son otra cosa que las coordenadas sobre la pantalla de vídeo de la proyección en perspectiva de un punto cualquiera X,Y,Z perteneciente al sólido originario.

Lo único que queda por hacer es la transformación necesaria, ya efectuada en otros programas, para llevar los dos ejes carte-

sianos de la página gráfica al centro de la pantalla y reducir los errores de redondeo. Las variables XV e YV calculadas por el programa comprenden ya esta última transformación: todo ello es efectuado de forma combinada por las líneas 170 y 180.

Al ejecutar el programa 3.1, cuyo principal procedimiento debería estar claro, aparecerá en la pantalla la imagen en perspectiva de un cubo. La figura que hay que dibujar, es decir, el cubo, está definida por medio de una serie de líneas DATA, desde la 470 a la 530. El programa también dibuja los tres ejes cartesianos del sistema de referencia originario x,y,z en el que ha sido definido el cubo. Esto es efectuado por las líneas 250-280 y sólo sirve para que la imagen resulte más comprensible. Los DATA relativos al dibujo de los tres ejes son los que figuran en las líneas 470-490.

Después siguen las coordenadas de los vértices del cubo (Figura 10). La línea 500 contiene la posición de los cuatro vértices de la cara anterior, la 510 la de la posterior. Hay cinco grupos de coordenadas por cada cara, ya que para unir cuatro puntos con cuatro segmentos es necesario completar todo el "giro" entre ellos (de ahí que el primer punto aparezca también como último, al que habrá que regresar para cerrar la superficie). Los cuatro segmentos que faltan para completar el cubo son trazados por las líneas 370-400, sirviéndose de los DATA que se hallan en las líneas 520 y 530.

Cuestión de puntos de vista

Las dimensiones de la imagen en perspectiva que obtenemos sobre la pantalla dependen de dos factores: el primero es la distancia del observador al origen, que hemos indicado con RH; el otro, la distancia "D" entre el observador y el plano de proyección. Bastará meditar brevemente acerca de la figura 5 para convencerse de que:

- si el observador se acerca al objeto (es decir, RH disminuye) las dimensiones de la proyección aumentarán;
- si lo que se acerca es el plano de proyección (y por tanto D disminuye) la imagen se hará más pequeña.

Podría parecer entonces que una disminución de RH se compensaría con un aumento adecuado de "D", pero en realidad esto es posible sólo en parte; veamos por qué. Consideremos las dos ecuaciones que permiten obtener XV e YV (líneas 170-180). En ellas "D" aparece como factor de un producto, así que podemos decir que tanto XV como YV son proporcionales al valor de "D"

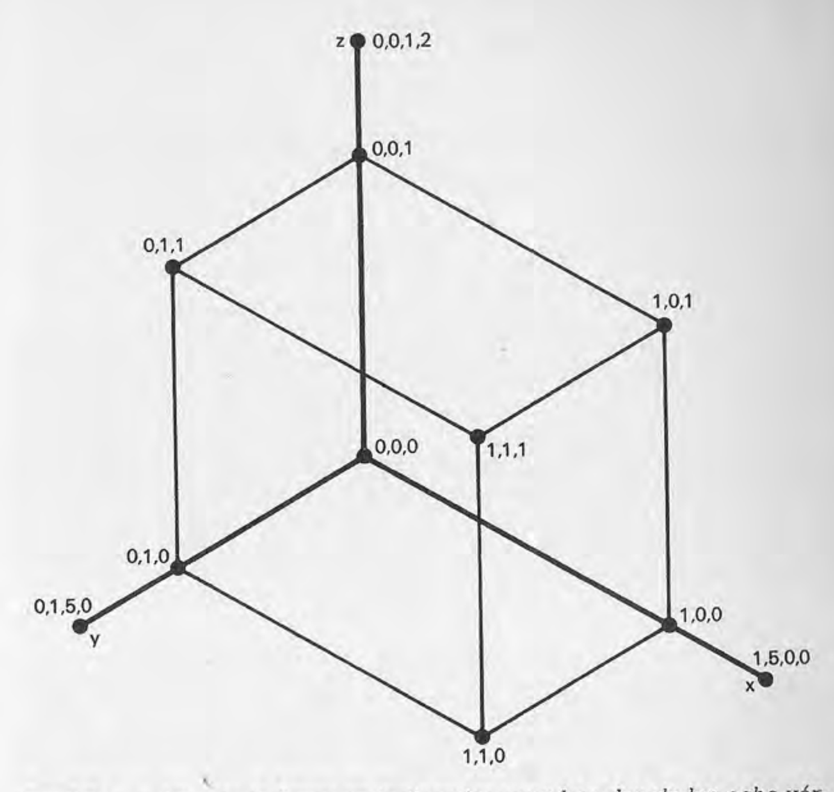


Figura 10.—En la figura se indican las coordenadas de los ocho vértices del cubo del programa 3.1, además de las de los puntos utilizados para trazar los ejes. El hecho de que el lado mida 1 no es esencial; al modificar "D" se le podrá asignar cualquier otra medida.

(aparte de las constantes utilizadas para llevar los ejes al centro de la pantalla). Pero el caso del parámetro RH, que aparece como término de una suma en la línea 160, es diferente: no sólo el resultado de la expresión (ZO) no es proporcional a RH, sino que su valor no influye sobre XO e YO. Podemos deducir de esto que, además de actuar sobre las dimensiones de la imagen, la variación de RH y de "D" provoca otros efectos:

- al ir acercándose al objeto (disminuye RH) éste nos aparecerá cada vez más deformado, del mismo modo que si estuviéramos utilizando una máquina fotográfica provista de un gran angular como objetivo;

— independientemente de esto, podemos modificar cuanto queramos las dimensiones de la imagen (pero no sus proporciones) al variar la distancia "D". Esta operación equivale a desplazar adelante y atrás una pantalla sobre la que estamos proyectando una diapositiva: la imagen se hará más grande o se reducirá manteniendo las proporciones inalteradas.

Si observa con un poco de atención el dibujo creado por el programa 3.1 notará precisamente una ligera distorsión de la imagen, como si estuviera observando el dibujo desde muy cerca. No se trata de un defecto: los parámetros del programa han sido elegidos precisamente para hacer más evidente este resultado; si quiere puede tratar de cambiarlos y ver qué ocurre.

Control total de la imagen

Después de haber aprendido la técnica, en muchas personas surgirá el deseo de hacer algún experimento. El programa 3.2 permite introducir a través del teclado los cuatro parámetros RH, TH, PH y D de manera que pueda obtenerse uno cualquiera de los efectos descritos hasta ahora.

```
10 REM *** PROGRAMA 3.2 ***
20 REM
30 REM CON NUESTRO PUNTO DE VISTA
40 REM
50 REM
60 XI=0:XD=319:YA=0:YB=199:FE=1.1
70 CX=INT(XD/2):CY=INT(YB/2)
80 RH=5:TH=30:PH=60:D=400
90 GOTO 560
100 REM
110 REM CALCULO COORDENADAS VISUALIZACION
120 REM
130 XO=-X*S1+Y*C1
140 YO=-X*C1*C2-Y*S1*C2+Z*S2
150 ZO=-X*S2*C1-Y*S2*S1-Z*C2+RH
160 XV=.5+CX+D*XO/ZO*FE
170 YV=.5+CY-D*YO/ZO
180 RETURN
190 REM
```



```
200 REM CLASIFICACION PUNTOS
210 REM
220 I(P)=X(P)<XI
230 D(P)=X(P)>XD
240 A(P)=Y(P)<YA
250 B(P)=Y(P)>YB
260 RETURN
270 REM
280 REM CALCULA LOS EXTREMOS
290 REM
300 GOSUB 220
310 V=0:IF I(1)*I(2)+D(1)*D(2)+A(1)*A(2)+
    B(1)*B(2)<>0 THEN 390
320 P=1:IF I(1)+D(1)+A(1)+B(1)<>0 THEN 350
330 P=2:IF I(2)+D(2)+A(2)+B(2)<>0 THEN 350
340 V=1:GOTO 390
350 IF I(P)<>0 THEN Y(P)=Y(1)+(XI-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XI:GOTO 300
360 IF D(P)<>0 THEN Y(P)=Y(1)+(XD-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XD:GOTO 300
370 IF A(P)<>0 THEN X(P)=X(1)+(YA-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=YA:GOTO 300
380 IF B(P)<>0 THEN X(P)=X(1)+(YB-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=YB:GOTO 300
390 RETURN
400 REM
410 REM PETICION PARAMETROS
420 REM
430 PRINT "♥PARAMETROS PARA LA VISUALIZACION"
440 PRINT "QQQDISTANCIA RHO (OBSERVADOR)...";RH
450 PRINT "QANGULO THETA (HORIZONTAL).....";TH
460 PRINT "QANGULO PHI (VERTICAL).........";PH
470 PRINT "QDISTANCIA D (PROYECCION)......";D
480 PRINT "SQQQQ";TAB(30);:INPUT RH
490 PRINT "Q";TAB(30);:INPUT TH
500 PRINT "Q";TAB(30);:INPUT PH
510 PRINT "Q";TAB(30);:INPUT D
520 RETURN
530 REM
540 REM PROGRAMA PRINCIPAL
```

```
550 REM
560 GOSUB 430:HIRES 1,0
570 S1=SIN(TH/180*PI):C1=COS(TH/180*PI)
580 S2=SIN(PH/180*PI):C2=COS(PH/180*PI)
590 RESTORE:READ NS
600 FOR SE=1 TO NS:FOR P=1 TO 2
610 READ X,Y,Z:GOSUB 130:X(P)=XV:Y(P)=YV:
    GOSUB 220
620 NEXT P
630 GOSUB 310:IF V THEN LINE X(1),Y(1),X(2),Y(2),1
640 NEXT SE
650 REM
660 POKE 198,0:REM A CERO BUFFER TECLADO
670 GET T$:IF T$="" THEN 670
680 PRINT "♥":NRM:PRINT "QQ TECLEA <RETURN> PARA
    CONTINUAR"
690 GET T$:IF T$="" THEN 690
700 IF T$=CHR$(13) THEN 560
710 END
720 REM
730 REM DEFINICION DE LA FIGURA
740 REM
750 DATA 14:REM No. SEGMENTOS
760 DATA 1,0,0,1,0,1,1,0,1,1,1,1,1,1,1,1,1,0
770 DATA 1,1,0,1,0,0,0,0,0,0,1,0,0,1,0,1,1
780 DATA 0,1,1,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0
790 DATA 1,0,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,1
800 DATA 1,0.33,0,0,0.33,1,1,0.66,0,0,0.66,1
READY
```

Para trabajar en la forma que resulta más habitual, los ángulos son expresados en grados; el programa se encargará de transformarlos en radianes antes de seguir calculando (líneas 570-580). Observará que en este nuevo programa ha sido añadida una subrutina de clipping (líneas 200-260 y 280-390) para que no sea necesario preocuparse de que los parámetros produzcan una imagen que se salga de los límites de la pantalla.

Varía también la manera de describir en las líneas DATA el sólido que va a ser dibujado (desde la 750 a la 800). Para ello se consideran uno por uno los segmentos que constituyen el objeto: la primera cantidad que aparece en los DATA indica su número, que se asigna posteriormente a la variable NS en la línea 590. Siguen luego dos grupos de tres coordenadas (X,Y,Z) por cada uno de los segmentos que hay que trazar. En el ejemplo hay 14 segmentos, así que las coordenadas son en total 84. Como ya sabrá de sobra es posible modificar los DATA para cambiar completamente el tipo de sólido que se va a visualizar.

Tal y como está, el programa dibuja nuevamente un cubo, pero esta vez sin los ejes cartesianos. Para no crear confusión acerca de la orientación del sólido visto en perspectiva se han añadido en su interior dos segmentos diagonales, que permitirán darse cuenta de los cambios de posición.

A la hora de asignar un valor a los cuatro parámetros, respondiendo a las peguntas formuladas, esta versión del programa (para el Commodore 64) permite confirmar el valor del parámetro introducido previamente al pulsar la tecla RETURN, sin tener necesidad de volver a introducir el mismo valor. Esto permite poder dibujar varias veces la misma figura con facilidad, con sólo cambiar los parámetros de los que se quiere observar su efecto.

Otras transformaciones

Al exponer la técnica de representación en perspectiva hemos utilizado algunos tipos de transformaciones obtenidas al aplicar a las coordenadas homogéneas matrices de 4×4. Damos a continuación algunas matrices útiles de las que no habíamos hablado hasta ahora por si alguien quiere hacer experimentos de gráficos computerizados más allá de los descritos en este libro.

Rotación alrededor del eje "y" en sentido opuesto al de las agujas de un reloj un ángulo "A":

$$\begin{vmatrix} COS(A) & 0 & -SIN(A) & 0 \\ 0 & 1 & 0 & 0 \\ SIN(A) & 0 & COS(A) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Simetría de un punto respecto del plano formado por los ejes "x" e "y":

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Simetría de un punto respecto del plano formado por los ejes "X" y "Z":

Variación de la escala en cada eje (respectivamente, A, B, C para los ejes x, y, z):

$$\begin{vmatrix} A & 0 & 0 & 0 \\ 0 & B & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Para completar este tema vamos a examinar las reglas para multiplicar una matriz de transformación (de 4×4 elementos) por un punto o por otra matriz. Esto permitirá conocer con todo detalle el cálculo de las coordenadas X0, Y0, Z0 que es efectuado en los programas 3.1 y 3.2. Para aplicar una matriz de transformación a un punto se efectúa el siguiente cálculo:

$$(X, Y, Z, 1) \begin{vmatrix} A & B & C & 0 \\ D & E & F & 0 \\ G & H & I & 0 \\ J & K & L & 1 \end{vmatrix} = \begin{matrix} (AX+DY+GZ+J, \\ BX+EY+HZ+K, \\ CX+FY+IZ+L, 1) \end{matrix}$$

En cambio, para multiplicar entre sí dos matrices de 4×4, en el caso de las transformaciones en secuencia, el desarrollo de la regla se hace excesivamente amplio. Daremos, por tanto, una definición algorítmica diciendo de qué manera se efectúa el producto sin tener que desarrollarlo por completo. Empezaremos por dibujar dos matrices, "A" y "B", y asignar un nombre a cada uno de los elementos:

$$\begin{vmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{vmatrix} \quad \begin{vmatrix} B11 & B12 & B13 & B14 \\ B21 & B22 & B23 & B24 \\ B31 & B32 & B33 & B34 \\ B41 & B42 & B43 & B44 \end{vmatrix}$$

Por cada uno de ellos el primer número indica la fila en la que se halla y el segundo la columna. Por ejemplo, A32 (que se

lee "a tres dos" y NO "a treinta y dos") pertenece a la tercera línea y a la segunda columna. Pues bien, cada elemento Cij de la matriz resultante (producto de las otras dos) se obtiene con la suma de los productos de todos los elementos de la fila "i" de la matriz "A" por todos los elementos de la columna "j" de la matriz "B".

Vamos a poner un ejemplo: el elemento C23 (línea 2 y columna 3 de la matriz resultante C) está dado por:

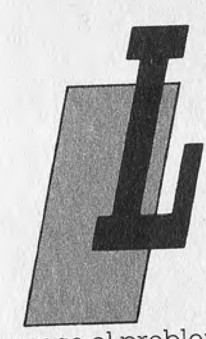
C23=A21*B13 + A22*B23 + A23*B33 + A24*B43

y esto para todos los restantes 15 elementos de la matriz C. Esta regla es general y válida para matrices de cualquier dimensión, con tal de que sea respetada la regla fundamental de que el número de líneas de la primera sea igual al número de columnas de la segunda. En los restantes casos no será posible efectuar el producto.

CAPITULO VII

TÉCNICAS GRÁFICAS AVANZADAS

El problema de las superficies escondidas

a técnica utilizada para dibujar objetos tridimensionales tiene una limitación evidente: las imágenes obtenidas no dan la sensación de representar un cuerpo sólido, sino que recuerdan más bien a las de su esqueleto, construido con alambres. Esto ocurre porque, al contrario de lo que pasa con los objetos reales, aquí están representadas también aquellas superficies que normalmente no son visibles debido a su posición. Así nace el problema de las superficies ocultas, que puede ser resuelto utilizando la técnica que ahora veremos.

Dado un "punto de vista" (perspectiva) cualquiera, lo primero que tenemos que hacer es fijar un criterio para establecer qué lados del objeto son visibles y cuáles no. Una vez hecha la clasificación, el programa deberá encargarse de trazar sólo aquellas superficies que corresponderían a la imagen del objeto que tendríamos si lo estuviéramos mirando realmente desde ese punto en concreto. Para experimentar esta nueva técnica nos referiremos al programa 3.3:

```
10 REM *** PROGRAMA 3.3 ***
20 REM
30 REM  DIBUJA UN OBJETO TRIDIMENSIONAL
40 REM ELIMINANDO LAS LINEAS NO VISIBLES
50 REM
60 REM
70 XI=0:XD=319:YA=0:YB=199:FE=1.1
```

```
80 CX=INT(XD/2):CY=INT(YB/2)
90 RH=20:TH=30:PH=60:D=200
100 GOTO 170
110 REM
120 REM PREPARACION LISTA DE LOS VERTICES
130 REM
140 READ NV:DIM V(NV,3)
150 FOR I=1 TO NV:FOR I1=1 TO 3
160 READ V(I,I1):NEXT I1,I
170 RETURN
180 REM
190 REM PREPARACION LISTA DE LAS SUPERFICIES
200 REM
210 READ NS:READ MP:DIM SU(NS,MP),NP(NS)
220 FOR I=1 TO NS:READ NP(I):FOR I1=1 TO NP(I)
230 READ SU(I,I1):NEXT I1,I
240 RETURN
250 REM
260 REM PREPARACION LISTA DE LAS NORMALES
270 REM
280 IF F=0 THEN DIM N(NS,3)
290 FOR I=1 TO NS
300 U1=V(SU(I,2),1)-V(SU(I,1),1)
310 U2=V(SU(I,2),2)-V(SU(I,1),2)
320 U3=V(SU(I,2),3)-V(SU(I,1),3)
330 V1=V(SU(I,3),1)-V(SU(I,1),1)
340 V2=V(SU(I,3),2)-V(SU(I,1),2)
350 V3=V(SU(I,3),3)-V(SU(I,1),3)
360 N(I,1)=U2*V3-V2*U3
370 N(I,2)=U3*V1-V3*U1
380 N(I,3)=U1*V2-V1*U2
390 NEXT I
400 RETURN
410 REM
420 REM PREPARACION LISTA DE LOS BORDES VISIBLES
430 REM
440 IF F=0 THEN DIM BV(NS*MP/2,2)
450 X1=RH*S2*C1:Y1=RH*S2*S1:Z1=RH*C2
460 B=1:FOR I=1 TO NS:P0=SU(I,1)
470 W1=X1-V(P0,1):W2=Y1-V(P0,2):W3=Z1-V(P0,3)
480 IF N(I,1)*W1+N(I,2)*W2+N(I,3)*W3<=0 THEN 590
```



```
490 FOR I1=2 TO NP(I)
500 P1=SU(I,I1-1):P2=SU(I,I1)
510 IF B=1 THEN 580
520 FOR I2=1 TO B-1
530 P3=BV(I2,1):P4=BV(I2,2)
540 IF (P1=P3 AND P2=P4) OR (P1=P4 AND P2=P3)
    THEN I2=9999
550 NEXT I2:IF I2>B THEN 580
560 BV(B,1)=P1:BV(B,2)=P2
570 B=B+1
580 NEXT I1
590 NEXT I
600 RETURN
610 REM
620 REM CALCULO COORDENADAS VISUALIZACION
630 REM
640 XO=-X*S1+Y*C1
650 YO=-X*C1*C2-Y*S1*C2+Z*S2
660 ZO=-X*S2*C1-Y*S2*S1-Z*C2+RH
670 XV=.5+CX+D*XO/ZO*FE
680 YV=.5+CY-D*YO/ZO
690 RETURN
700 REM
710 REM TRAZADO DE LA FIGURA
720 REM
730 FOR I=1 TO B-1:FOR P=1 TO 2
740 X=V(BV(I,P),1):Y=V(BV(I,P),2):Z=
    V(BV(I,P),3)
750 GOSUB 640:X(P)=XV:Y(P)=YV:GOSUB 830
760 NEXT P
770 GOSUB 920:IF V=1 THEN LINE X(1),Y(1),
    X(2),Y(2),1
780 NEXT I
790 RETURN
800 REM
810 REM CLASIFICACION PUNTOS
820 REM
830 I(P)=X(P)<XI
840 D(P)=X(P)>XD
850 A(P)=Y(P)<YA
860 B(P)=Y(P)>YB
```

```
870 RETURN
880 REM
890 REM CALCULA LOS EXTREMOS (CLIPPING)
900 REM
910 GOSUB 830
920 V=0:IF I(1)*I(2)+D(1)*D(2)+A(1)*A(2)+
    B(1)*B(2)<>0 THEN 1000
930 P=1:IF I(1)+D(1)+A(1)+B(1)<>0 THEN 960
940 P=2:IF I(2)+D(2)+A(2)+B(2)<>0 THEN 960
950 V=1:GOTO 1000
960 IF I(P)<>0 THEN Y(P)=Y(1)+(XI-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XI:GOTO 910
970 IF D(P)<>0 THEN Y(P)=Y(1)+(XD-X(1))*(Y(2)-
    Y(1))/(X(2)-X(1)):X(P)=XD:GOTO 910
980 IF A(P)<>0 THEN X(P)=X(1)+(YA-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=YA:GOTO 910
990 IF B(P)<>0 THEN X(P)=X(1)+(YB-Y(1))*(X(2)-
    X(1))/(Y(2)-Y(1)):Y(P)=YB:GOTO 910
1000 RETURN
1010 REM
1020 REM PETICION PARAMETROS
1030 REM
1040 PRINT "♥PARAMETROS PARA LA VISUALIZACION"
1050 PRINT "QQQQ DISTANCIA RHO (OBSERVADOR)...";RH
1060 PRINT "Q ANGULO THETA (HORIZONTAL).......";TH
1070 PRINT "Q ANGULO PHI (VERTICAL)...........";PH
1080 PRINT "Q DISTANCIA D (PROYECCION)........";D
1090 PRINT "5QQQQQ";TAB(29);:INPUT RH
1100 PRINT "Q";TAB(29);:INPUT TH
1110 PRINT "Q";TAB(29);:INPUT PH
1120 PRINT "Q";TAB(29);:INPUT D
1130 RETURN
1140 REM
1150 REM PROGRAMA PRINCIPAL
1160 REM
1170 DIM I(2),D(2),A(2),B(2)
1180 GOSUB 1040:HIRES 1,0
1190 S1=SIN(TH/180*PI):C1=COS(TH/180*PI)
1200 S2=SIN(PH/180*PI):C2=COS(PH/180*PI)
1210 IF F=1 THEN 1240
1220 GOSUB 140:REM LISTA VERTICES
```



```
1230 GOSUB 210:REM LISTA SUPERFICIES
1240 GOSUB 280:REM LISTA NORMALES
1250 GOSUB 440:REM LISTA BORDES VISIBLES
1260 GOSUB 730:REM TRAZADO FIGURA
1270 REM
1280 F=1:POKE 198,0:REM PUESTA A CERO DEL
     BUFFER DEL TECLADO
1290 GET T$:IF T$="" THEN 1290
1300 PRINT "♥":NRM:PRINT "QQQ TECLEA <RETURN>
     PARA CONTINUAR"
1310 GET T$:IF T$="" THEN 1310
1320 IF T$=CHR$(13) THEN 1180
1330 PRINT:PRINT "QQQ OK,ADIOS":END
1340 REM
1350 REM DESCRIPCION DE LA FIGURA
1360 REM
1370 DATA 5,2,5,-4,-2,5,-4,-2,-5,-4,2,-5,
     -4,0,0,4
1380 DATA 5,5,4,1,5,4,1,4,3,5,2,3
1390 DATA 4,2,5,1,2,4,4,5,3,4,5,1,4,3,2,1
READY
```

La figura que vamos a dibujar, una pirámide con base rectangular, está codificada en el grupo de instrucciones DATA que se hallan al final del programa (líneas 1370-1390). El procedimiento que estamos examinando requiere que sean proporcionadas las posiciones de todos los vértices del objeto y la definición de cada una de sus superficies en relación a ellos. Refiriéndonos a la figura 1, las posiciones de los vértices son:

vértice número 1=2,5,–4
vértice número 2=–2,5,–4
vértice número 3=–2,–5,–4
vértice número 4=2,–5,–4
vértice número 5=0,0,4

Cada superficie se define posteriormente por medio de la secuencia de los vértices que la delimitan. Según el procedimiento empleado, resulta esencial que éstos sean puestos por orden en sentido contrario al de las agujas del reloj con relación a un observador que mire la superficie desde el exterior del sólido. Además, una superficie definida, por ejemplo, por cuatro puntos será indicada como si lo estuviera por cinco, ya que al efectuar el dibujo el último punto deberá unirse con el primero. Una vez pre-

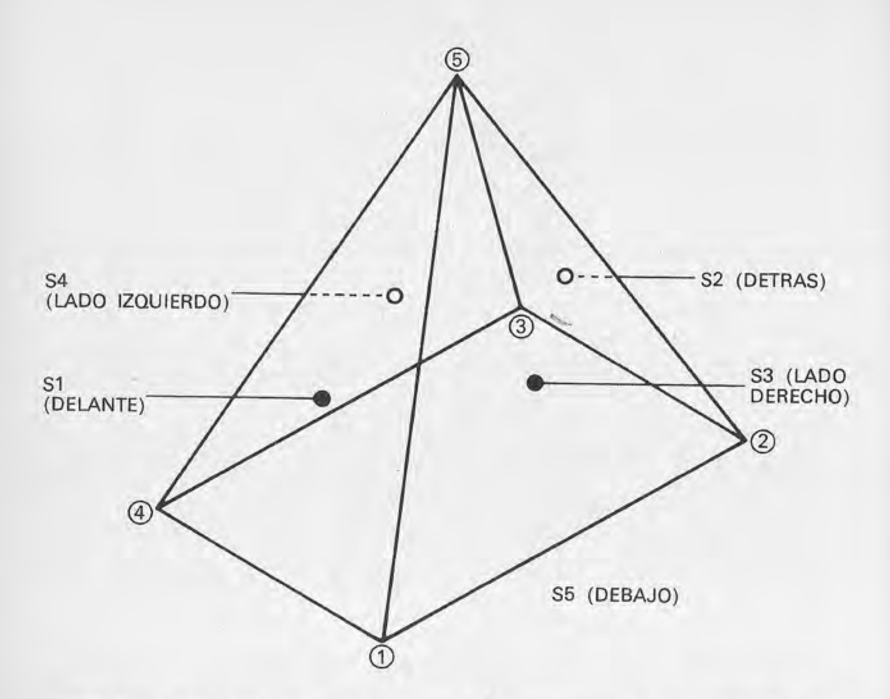


Figura 1.—El sólido que vamos a dibujar es una pirámide con base rectangular, traspasada por el eje "z" de forma que el origen se halle a media altura (aquí los ejes no están trazados). Las coordenadas de los vértices y la definición de las superficies, desde S1 hasta S5, están contenidas en las líneas DATA del programa 3.3.

cisados estos detalles, explicaremos de qué manera están organizadas las líneas DATA:

— número de vértices, coordenadas X,Y,Z del vértice número 1, coordenadas X,Y,Z del vértice número 2, y así hasta el último vértice (línea 1370);
— número de superficies, número máximo de vértices por superficie (línea 1380);
— número de vértices de la superficie número 1, primer vértice de la superficie número 1, segundo vértice de la superficie número 1, etc. (línea 1380);
— número de vértices de la superficie número 2, etc. (también en la línea 1380);
— sigue así hasta la última superficie, la número 5 (línea 1390).

El programa 3.3 pone en práctica la técnica de las superficies ocultas, que representa en cierto modo la síntesis de todas las que hemos visto hasta ahora. Después de las mismas inicializaciones de siempre (líneas 70-90) el control pasa a la línea 1170, donde comienza el programa principal. El motivo por el cual están dimensionados los vectores I, D, A, B utilizados luego en la subrutina habitual de clipping, es el de ahorrar espacio en memoria. En efecto, si no se especifica, el BASIC dimensiona automáticamente cada vector asignándole once elementos (de 0 a 10); al escribir programas complejos es bueno acostumbrarse a no derrochar las reservas del ordenador.

A continuación se ejecuta la subrutina, ya vista en el programa 3.2, que solicita los parámetros (líneas 1020-1130), luego se visualiza en la pantalla la página gráfica (línea 1180).

A partir de los datos proporcionados por el usuario, las líneas 1190 y 1120 calculan el valor de las habituales variables S1, S2, C1, C2. Recuerde que los datos están expresados en los comunes grados sexagesimales. El salto condicionado a la línea 1210 sirve para ejecutar una parte del procedimiento tan sólo la primera vez, ahorrando así tiempo de procesamiento cuando se repite el cálculo cambiando exclusivamente el punto de observación (pero no la figura que se quiere trazar).

Identificación de las superficies visibles

El método para seleccionar las superficies del sólido que no quedan ocultas para el observador se basa en una serie de subrutinas, llamadas a partir de la línea 1220, y procede de esta manera:

1. Partiendo de la descripción del objeto contenida en las líneas DATA se prepara una lista de vértices (líneas 120-170). Esta lista se construye en un vector (o matriz) de dos dimensiones llamado V().
2. Basándose siempre en los DATA se rellenan otros dos vectores, SU() y NP() (líneas 190-240). El primero es también de dos dimensiones y contiene una lista de los vértices que delimitan cada superficie (en sentido opuesto al de las agujas del reloj); para cada una de ellas, el segundo vector (un vector unidimensional) indicará el número de vértices que la componen.
3. Disponiendo de las listas obtenidas en los pasos 1 y 2 se utiliza un método de cálculo que, recurriendo al concepto matemático-geométrico de vector, determina la orientación de cada superficie localizando la dirección de una normal

(es decir, de una línea perpendicular a ella). Esto se lleva a cabo en las líneas 260-400.

4. Ya podemos combinar las informaciones acerca de la orientación de cada superficie con las de la posición del observador, para determinar por último si cada una de ellas es visible o no desde ese punto de vista en concreto (líneas 420-600).
5. Ahora no queda más que someter los vértices de las superficies visibles al procedimiento que ya conocemos de representación en perspectiva (líneas 620-690) y unirlos luego con segmentos a fin de obtener los contornos de aquella superficie (líneas 710-790).

Lo que se obtiene es una fiel representación de cómo vería un observador el objeto al mirar desde esa posición. El método utilizado en el programa 3.3 funciona correctamente sólo si se aplica a sólidos convexos, es decir, a aquellos cuyas superficies no forman nunca ángulos externos menores que 180 grados (o, en otras palabas, que no tienen "entradas"). Todo el proceso es bastante complejo, así que convendrá seguirlo deteniéndose en los detalles que necesitan que profundicemos más.

Primer paso: lista de los vértices

La matriz que contiene la lista de los vértices del sólido aparece en la figura 2. La subrutina que la rellena (líneas 140-170) a partir de los DATA no necesita particulares comentarios.

Segundo paso: lista de las superficies

La figura 3 muestra el vector SU() que describe cada superficie mediante la lista de vértices que la delimitan. Aparece también el vector que contiene el número de vértices para cada una de ellas NP(). La subrutina que los llena se halla en las líneas 210-240.

Tercer paso: lista de las normales (orientación de las superficies)

Para determinar de qué manera están orientadas las superficies que delimitan el sólido que va a ser representado es necesario utilizar un nuevo concepto: el de vector. Podemos describir un vector como un segmento orientado, es decir, que posee un sentido determinado, que une dos puntos ya sea que estén en un

PRIMER INDICE	SEGUNDO INDICE X	Y	Z
VERTICE N.º 1	2	5	−4
VERTICE N.º 2	−2	5	−4
VERTICE N.º 3	−2	−5	−4
VERTICE N.º 4	2	−5	−4
VERTICE N.º 5	0	0	4

LISTA DE LOS VERTICES: VECTOR V( )

Figura 2.—A cada vértice se le asigna un número de orden, en este caso de 1 a 5. El vector $\vec{V}()$ contiene las coordenadas X,Y,Z.

PRIMER INDICE:	SEGUNDO INDICE: VERTICES QUE HAY QUE UNIR					INDICE:	
SUPERFICIE N.º 1	5	5	4	1		N.º VERTICES SUP. 1	4
SUPERFICIE N.º 2	3	5	2	3		N.º VERTICES SUP. 2	4
SUPERFICIE N.º 3	2	5	1	2		N.º VERTICES SUP. 3	4
SUPERFICIE N.º 4	4	5	3	4		N.º VERTICES SUP. 4	4
SUPERFICIE N.º 5	1	4	3	2	1	N.º VERTICES SUP. 5	5

LISTA DE LOS VERTICES: VECTOR SU ( )

NUMERO DE VERTICES POR CADA SUPERFICIE: VECTOR NP ( )

Figura 3.—Vectores que describen las superficies del sólido. Los vértices deben ponerse en la lista procediendo en sentido contrario al de las agujas de un reloj (mirando la superficie desde el exterior).

plano o en el espacio. También en este caso no nos es posible profundizar sobre el tema, pero es conveniente saber que se puede identificar un vector a partir de las coordenadas de sus puntos terminales. Vayamos a la figura 4: considerando como extremos los puntos A(XA,YA,ZA) y B(XB,YB,ZB) resulta que el vector $\vec{AB}$ está definido por la terna (XB-XA,YB-YA,ZB-ZA).

Del mismo modo que ocurre con los números, también los vectores pueden sumarse, sustraerse, multiplicarse o dividirse entre sí. Sin entrar en detalles sobre la teoría de vectores vamos a considerar un tipo particular de multiplicación entre ellos, llamado producto vectorial. Partiendo, por ejemplo, de los vectores $\vec{V1}$=(X1,Y1,Z1) y $\vec{V2}$=(X2,Y2,Z2) su producto vectorial se define como:

$$\vec{V1} \times \vec{V2} = (Y1*Z2-Z1*Y2, Z1*X2-X1*Z2, X1*Y2-Y1*X2)$$

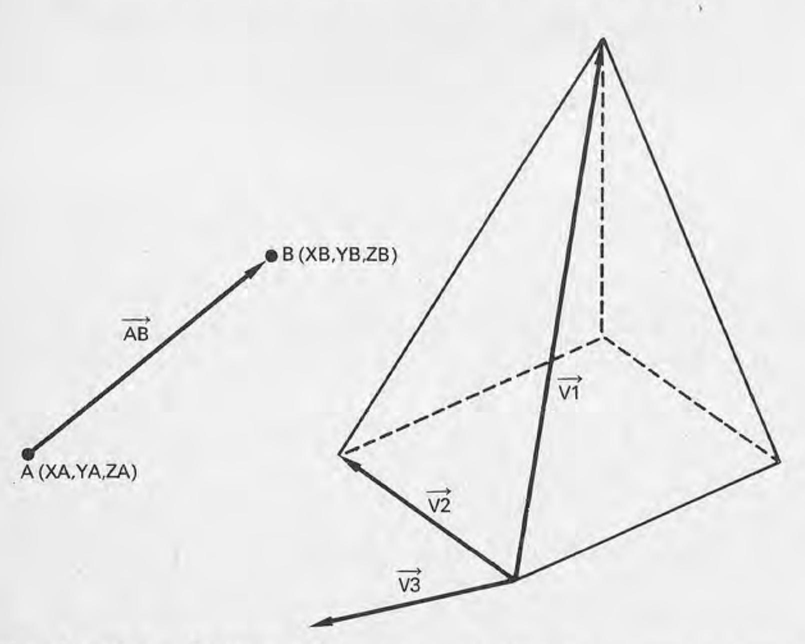


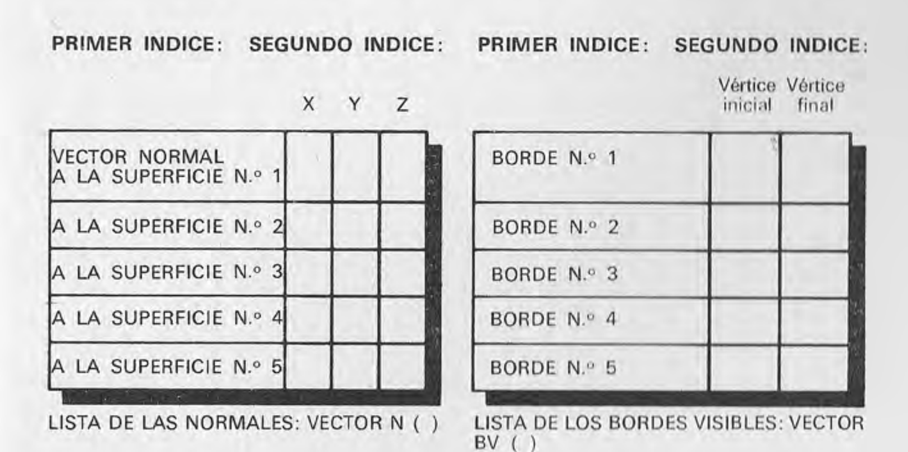
Figura 4.—Un vector $\vec{AB}$ puede ser identificado a partir de las coordenadas de sus dos extremos. El producto vectorial entre dos vectores, por ejemplo $\vec{V1}$ y $\vec{V2}$, representa también otro vector, que siempre será perpendicular al plano en el que se hallan $\vec{V1}$ y $\vec{V2}$. Al escoger los dos vectores de partida como está indicado, $\vec{V3}$ señalará siempre hacia el exterior del sólido. Los vectores $\vec{U}$,$\vec{V}$,$\vec{N}$ de la subrutina 280 corresponden a $\vec{V1}$, $\vec{V2}$ y $\vec{V3}$ en la figura. Esto mismo se repite para cada superficie examinada.

No hay que olvidar que el producto vectorial de dos vectores, es decir, la terna resultante, representa otro vector.

Lo que nos interesa es una propiedad geométricia de este producto: el vector que se obtiene resulta siempre perpendicular al plano sobre el que se hallan $\vec{V1}$ y $\vec{V2}$ (los dos vectores de partida) y su sentido se halla "girando" $\vec{V1}$ hacia $\vec{V2}$, como si fuera un destornillador, y viendo hacia dónde iría. La situación está ilustrada en la figura 4, a partir de la cual podemos darnos cuenta de que el vector $\vec{V3}$ (que resulta del producto vectorial de $\vec{V1}$ y $\vec{V2}$) puede ser utilizado para indicar la orientación de la superficie. Se dice que $\vec{V3}$ es una normal de la superficie, una línea perpendicular a ella.

La lista de las normales es formada por la subrutina comprendida entre las líneas 280–400. A cada una de las superficies del sólido se asocian dos vectores $\vec{U}$ y $\vec{V}$, correspondientes a V1 y $\vec{V2}$ de la figura 4, e identificados en el programa con las ternas (U1,U2,U3) y (V1,V2,V3). A partir de éstos se calcula la terna correspondiente a la normal de esa superficie (efectuando el producto vectorial de $\vec{U}$ y $\vec{V}$, líneas 360-380), que es memorizada en el vector N(). Al final del ciclo el vector contiene la lista de las normales a cada superficie.

Al observar la subrutina puede notarse que los dos vectores que hay que multiplicar se eligen siempre de forma que el segundo esté orientado de forma que su rotación sea opuesta a la



PRIMER INDICE:	SEGUNDO INDICE: X	Y	Z
VECTOR NORMAL A LA SUPERFICIE N.º 1			
A LA SUPERFICIE N.º 2			
A LA SUPERFICIE N.º 3			
A LA SUPERFICIE N.º 4			
A LA SUPERFICIE N.º 5			

LISTA DE LAS NORMALES: VECTOR N ( )

PRIMER INDICE:	SEGUNDO INDICE: Vértice inicial	Vértice final
BORDE N.º 1		
BORDE N.º 2		
BORDE N.º 3		
BORDE N.º 4		
BORDE N.º 5		

LISTA DE LOS BORDES VISIBLES: VECTOR BV ( )

Figura 5.—La lista de las normales a cada superficie es realizada por una subrutina específica a partir del vector $\vec{SU}$(). En base a esto se halla una lista de los bordes visibles, que indica los vértices iniciales y finales de cada segmento que hay que trazar.

del primero según se observa la superficie desde el exterior. Tal elección puede ser efectuada con seguridad, ya que nos hemos ocupado de hacer que en la lista de las superficies los vértices estén precisamente en el orden obtenido recorriéndolos en sentido opuesto al de las agujas de un reloj. Gracias a esta precaución, cada normal estará siempre dirigida hacia el exterior del sólido.

Cuarto paso: test de visibilidad de cada superficie

La última lista que hay que determinar es la de los bordes de las superficies visibles. El vector $\overline{BV}$() destinado a contenerlo es dimensionado en la línea 440, con un criterio bastante arbitrario, pero tal que asigna un número de elementos suficientes para este fin.

Para determinar si una superficie es visible recurrimos a otra propiedad de los vectores. Se trata esta vez del llamado producto escalar, que es otro tipo de multiplicación entre vectores distinto del producto vectorial y que da como resultado un número, no un vector. Tampoco en este caso nos es posible profundizar mucho en la teoría matemática, así que diremos solamente que, dados los vectores $\overrightarrow{V1}$ y $\overrightarrow{V2}$, su producto escalar toma un valor positivo distinto de cero sólo en el caso de que el ángulo entre ambos sea menor que 90 grados. Esta importante propiedad nos permite distinguir aquellas superficies orientadas hacia el observador de las que, en cambio, no lo están, como aparece claramente ilustrado en la figura, 6.

En la línea 450 se calcula la posición del observador, asignada a las variables X1, Y1 y Z1 por medio de las reglas trigonométricas vistas en otras ocasiones. Se comienza así a contar los bordes visibles con la variable "B", que parte del valor 1 (línea 460). Un ciclo FOR...NEXT (460-590) examina cada superficie, considerando el vértice número PØ (línea 460). La 470 determina la terna correspondiente a un vector $\vec{W}$ = (W1,W2,W3), que parte del vértice PØ y está dirigido hacia el observador.

Considerando dos vectores genéricos $\overline{V1}$=(X1,Y1,Z1) (que no hay que confundir con las coordenadas del observador) y $\overrightarrow{V2}$=(X2,Y2,Z2), la teoría dice que su producto escalar está dado por:

$$\overline{V1}*\overline{V2}=X1*X2+Y1*Y2+Z1*Z2$$

A diferencia del producto vectorial, el producto escalar NO es un vector, sino un simple número. La línea 480 calcula precisamente el producto escalar entre la normal y $\vec{W}$; si no resulta ma-

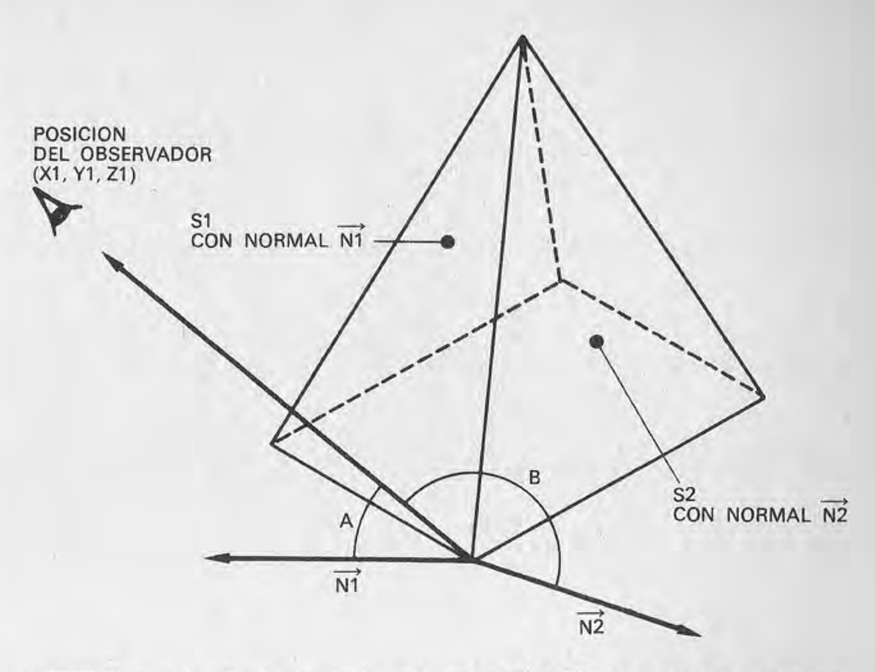


Figura 6.—En la figura los vectores $\overline{N1}$ y $\overline{N2}$ son normales, respectivamente, a las superficies S1 y S2 de la pirámide. Consideremos el vector $\overline{W}$, que sale de un punto cualquiera de la superficie y está dirigido hacia la posición del observador. La superficie S1 resulta visible porque el ángulo A entre $\overline{W}$ y $\overline{N1}$ es menor que 90 grados, mientras que S2 permanece oculta porque el ángulo B entre $\overline{W}$ y $\overline{N2}$ es, en cambio, mayor que 90 grados.

yor que cero la superficie es desechada, ya que no resulta visible desde esa posición.

En caso contrario, el bucle con variable I1 que se inicia en la 490 incluye en el vector $\vec{B}$() los vértices de la superficie, tomándolos dos a dos y formando así una lista de segmentos visibles (o bordes visibles). En esta fase está también previsto un control para evitar poner en lista dos veces el mismo segmento, como ocurriría en el caso del borde entre dos superficies visibles. Este control está seguido por el bucle con índice I2 que se inicia en la línea 520.

Quinto paso: dibujo en perspectiva

Ahora no queda más que dibujar la imagen en perspectiva del sólido. Para alcanzar este objetivo, los extremos de cada bor-

de visible son sometidos al ya conocido procedimiento de proyección tridimensional, el mismo que hemos utilizado en los programas 3.1 y 3.2 (subrutina 640-690). Para hacer más completo y "profesional" el procedimiento, se utiliza también una subrutina de clipping (líneas 810-870 y 890-1000), con el fin de no limitar de ninguna manera la posición del observador. El resultado de toda la operación será dibujado luego en la pantalla.

Con esto concluye la descripción de la técnica de eliminación de las superficies no visibles. Por último, teniendo en cuenta la complejidad del procedimiento y las consiguientes dificultades que pueden obstaculizar su comprensión, hemos incluido una lista completa de las variables utilizadas en el programa 3.3.

Lista de las variables del programa 3.3

XI,XD,YA,YB=límites a la izquierda, a la derecha, por arriba y por abajo de la pantalla.

CX,CY=coordenadas del centro de la pantalla.

RH=distancia del observador al origen.

TH=ángulo θ (theta).

PH=ángulo ϕ (phi).

D=distancia del plano de proyección al origen.

NV=número de vértices del sólido.

V()=lista de los vértices (vector de dos dimensiones).

X,Y,Z=coordenadas de los vértices del objeto.

XO,YO,ZO=coordenadas respecto del observador.

S1,C1=seno de θ (theta) y coseno de θ (theta).

S2,C2=seno de ϕ (phi) y coseno de ϕ (phi).

XV,YV=coordenadas de visualización de un punto.

I(),D(),A(),B()=indicadores que señalan si un punto se halla fuera de uno de los límites de la pantalla.

X(),Y()=puntos extremos del segmento que hay que someter a clipping.

V=distinto de cero si un segmento es dibujable en pantalla.

P=índice de los puntos de un segmento para la subrutina de clipping.

P$=parámetro solicitado.

I=índice genérico.

MP=máximo número de puntos (vértices) entre todas las superficies.

SU()=lista de las superficies (vector de dos dimensiones).

NS=número total de superficies del sólido.

NP()=número de los puntos de cada superficie (vector de una dimensión).

I1=índice genérico.

N()=lista de las normales (vector de dos dimensiones).

U1,U2,U3=coordenadas del vector U para el cálculo de la normal.

V1,V2,V3=coordenadas del vector V para el cálculo de la normal.

BV()=lista de los bordes visibles.

X1,Y1,Z1=coordenadas del observador.

PØ=número del primer punto de una superficie.

P1,P2=extremos de un borde visible recién determinado.

P3,P4=extremos de un borde visible ya en lista.

T$=tecla apretada al final del dibujo.

F=indicador que señala que los vectores ya han sido dimensionados.

Conclusión

Todas las técnicas que hemos examinado en este libro van acompañadas de programas demostrativos, escritos de forma que animen a modificarlos y experimentar con ellos tanto en el Commodore 64 como en otros ordenadores personales. En particular, los objetos del programa (las figuras geométricas) son la mayoría de las veces fácilmente definibles de nuevo, ya que están descritos por medio de líneas DATA. El hecho de modificar los programas, las figuras codificadas en ellos o desarrollar nuevas técnicas constituirá una valiosa ayuda para entender plenamente las bases de los gráficos computerizados.

APENDICE

En muchos de los listados aparecen dentro de las instrucciones PRINT símbolos gráficos, de algunos de los cuales hemos explicado ya el significado. Las siguientes tablas son un resumen de aquellos que afectan al control de la impresión y del color.

Símbolo	*Efecto*
□	Espacio
]	Cursor a la derecha
I	Cursor a la izquierda
Q	Cursor hacia abajo
●	Cursor hacia arriba
S	Cursor al punto HOME (arriba a la izquierda)
♥	Vaciar la pantalla
R	Inversión de contraste (negro sobre blanco)
—	Vuelta al contraste normal (blanco sobre negro)
T	Supresión de carácter
I	Inserción de carácter

Símbolo	Cód. ASCII	Efecto
	144	negro
	5	blanco
	28	rojo
	159	turquesa
	156	púrpura
	30	verde
	31	azul oscuro
	158	amarillo
	129	anaranjado
	149	pardo
	150	rosa
	151	gris oscuro
	152	gris medio
	153	verde claro
	154	azul claro
	155	gris claro

BIBLIOGRAFIA

Diseño de gráficos y videojuegos. Tratamiento en tres dimensiones.
Angel y Jones, 1985. *Ed. Anaya Multimedia.*

Programación en BASIC: un método práctico.
Dachslager y Zucker. *Ed. Anaya Multimedia.*

Programación avanzada en BASIC.
Bishop. *Ed. Anaya Multimedia.*

El libro del IBM PC, XT, AT.
L. E. Frenzel Jr. / L. E. Frenzel III, 1985. *Ed. Anaya Multimedia.*

Cómo programar su COMMODORE 64 1 - BASIC, gráficos, sonido.
F. Montell, 1985. *Ed. Paraninfo.*

Cómo programar un COMMODORE 64 2 - Lenguaje máquina, E/S, periféricos.
F. Montell, 1985. *Ed. Paraninfo.*

El libro del Apple IIc.
P. Lieberman. Ed. Anaya Multimedia.

Tratamiento de textos con BASIC.
G. Quaneaux, 1985. *Ed. Paraninfo.*

Informática para no avanzados.
G. Willmott, 1985. *Ed. Deusto.*

Programación del COMMODORE 64.
Gibbson. *Ed. Anaya Multimedia.*

Claves para el COMMODORE 64.
D. Gean David. *Ed. Elisa.*

El descubrimiento del COMMODORE 64.
David. *Ed. Elisa.*

COMMODORE 64 para todos.
Boisgontier Brebion Foucault. *Ed. Elisa.*

MSX. Guía del usuario.
Hoffman. *Ed. McGraw-Hill.*

Los ordenadores. Fundamentos y sistemas.
J. C. Giarratano, 1984. *Ed. Díaz de Santos.*

Conceptos actuales sobre la tecnología de los ordenadores.
J. C. Giarratano, 1984. *Ed. Díaz de Santos.*

Claves para el Apple II.
Breaud-Pouliquen. *Ed. Elisa.*

Claves para el ZX-Spectrum.
J. François Séhan. *Ed. Elisa.*

El BASIC de la A a la Z.
J. Boisgontier. *Ed. Elisa.*

COMMODORE 64 para todos.
J. Boisgontier. *Ed. Elisa.*

102 Programas para COMMODORE 64.
J. Deconchat. *Ed. Elisa.*

102 Programas para su APPLE.
Desconchat. *Ed. Elisa.*

102 Programas para XZ81 y Spectrum.
J. Deconchat. *Ed. Elisa.*

ZX Spectrum para todos.
Boisgontier. *Ed. Elisa.*

El Apple y sus ficheros.
J. Boisgontier. *Ed. Elisa.*

El Commodore 64 y los niños.
Solomon. *Ed. Noray.*

Microordenadores y cassettes.
M. Salem. *Ed. Noray.*

Profundizando en el ZX-Spectrum.
D. Jones. *Ed. Noray.*

Diccionario de informática inglés-español-francés.
G. A. Mania, 1985. *Ed. Paraninfo.*

Diccionario del BASIC.
Willie Hart, 1985. *Ed. Paraninfo.*

Diccionario de informática inglés-español. Glosario de términos informáticos.
Olivetti, 5 edic., 1984. *Ed. Paraninfo.*

Cómo programar ordenadores personales.
R. Farrando, 1985. *Ed. Marcombo.*

Diccionario de informática.
Masson, 2 edic., 1985. *Ed. Masson.*

Glosario de computación.
Alan Freedman, 1984. *Ed. McGraw-Hill.*

Diccionario del BASIC.
A. Lien. *Ed. Elisa.*

BIBLIOTECA BASICA INFORMATICA

INDICE GENERAL

10 **Sistemas operativos y software de base**
Qué son, para qué sirven. Unos desconocidos muy importantes.

11 **Sistema operativo CP/M**
Uno de los sistemas operativos para microprocesadores de 8 bits de mayor difusión en el mercado.

12 **MS-DOS: el estándar de IBM**
Sistema operativo para el microprocesador de 16 bits 8088, adoptado por el IBM-PC.

13 **Paquetes de aplicaciones. Software "pret a porter"**
Características y peculiaridades de los más importantes paquetes de aplicaciones.

14 **VisiCalc: una buena hoja de cálculo**
Interioridades y manejo de una de las hojas de cálculo más usadas.

15 **Dibujar con el ordenador**
Profundizando en una de las facetas útiles y divertidas que nos ofrecen los ordenadores.

16 **Tratamiento de textos... para escribir con el ordenador**
Cómo convertir su ordenador en una máquina de escribir con memoria y todo tipo de posibilidades.

17 **Diseño de juegos**
Particularidades características de esta aplicación de los ordenadores.

18 **LOGO: la tortuga inteligente**
Un lenguaje conocido por su «cursor gráfico», la tortuga, y sus aplicaciones pedagógicas al alcance de su mano.

19 **BASIC y tratamiento de imágenes**
Todo lo que en **¡Y llegó el BASIC!** no se pudo ver sobre las imágenes y gráficos en el BASIC.

20 **Bancos de datos (I)**
Peculiaridades de una de las aplicaciones de los ordenadores más interesantes, y que más dinero mueven.

21 **Bancos de datos (II)**
Profundizando en sus características.

22 **Paquetes integrados: Lotus 1-2-3 y Simphony**
Estudio de dos de los paquetes integrados (Hoja de cálculo + base de datos + ...) más conocidos.

23 **dBASE II y dBASE III**
Cómo aprovechar las dos versiones más recientes de esta importante base de datos.

24 **Los ordenadores uno a uno**
Un amplio y completo estudio comparativo.

25 **Cálculo numérico en BASIC**
Una aplicación especializada a su disposición.

26 **Multiplan**
Cómo hacer uso de este moderno paquete de aplicaciones.

27 **FORTRAN y COBOL**
Dos lenguajes muy especializados y distintos.

28 **FORTH: anatomía de un lenguaje inteligente**
Principales características de un lenguaje moderno, flexible y de amplio uso, en la robótica.

29 **Cómo realizar nuestro propio banco de datos**
Conocimientos necesarios para poder fabricar un banco de datos a nuestro gusto y medida.

30 **Los paquetes integrados uno a uno**
Todos los que usted puede encontrar en el mercado.

NOTA: **Ingelek, S. A. se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.**

NOTAS

Quién puede afirmar que no se ha quedado nunca atónito frente a una pantalla de ordenador admirando las increíbles imágenes que reproducía?

Las técnicas de dibujo computerizado alcanzan ya a muchos campos: gráficas de gestión, diseño industrial, películas, vídeo-juegos... Ahora bien, ¿puede entrar aquí en acción nuestro ordenador? Por supuesto que sí; sólo hace falta que le instruyamos adecuadamente, como siempre.

Así, deberemos aprender las técnicas que nos permitirán dibujar sobre la pantalla puntos, líneas, figuras y luego aprender a moverlas, bien sea mediante traslaciones o rotaciones. Una vez los dominemos podremos dedicarnos a representar objetos tridimensionales en todas las perspectivas que queramos, haciendo aparecer o no las líneas ocultas.

Todo esto es lo que veremos en el libro, partiendo de una teoría completa, explicada desde el principio para que nadie "se pierda", y con la ayuda de numerosos programas, prácticos y flexibles, escritos en BASIC.

395 pts.
(incluido IVA)